UNIVERSIDADE DE LISBOA

FACULDADE DE CIÊNCIAS

DEPARTAMENTO DE INFORMÁTICA

SOLUÇÃO MOBILE – BANKING, TRANSAÇÕES

Henrique André da Silva Candeias

MESTRADO EM ENGENHARIA INFORMÁTICA
Especialização em Engenharia de Software

Trabalho de Projeto orientado por:
Prof. Doutor João Ferreira
Eng. Ricardo Nogueira

2018

Agradecimentos

Não poderia deixar de agradecer ao meu coordenador do trabalho de projeto, Professor Doutor João Diogo Silva Ferreira, pela confiança, motivação, empenho e disponibilidade dispensada para a conclusão do trabalho de projeto.

Ao Engenheiro Ricardo Nogueira, *manager* da *Accenture*, pela disponibilidade, informações e conselhos facultados no desenvolvimento do projeto.

Aos meus colegas e amigos, pelo apoio, confiança, amizade e companheirismo demonstrado ao longo dos últimos anos.

À Faculdade de Ciências da Universidade de Lisboa, em especial aos docentes e colegas, que contribuíram para o conhecimento académico, pessoal e social, no decorrer dos últimos cinco anos.

À minha família, um especial e eterno agradecimento, pois sem eles não seria possível a concretização deste curso.

Os meus sinceros agradecimentos!

"Para vencerem na era digital, os Bancos devem construir um conjunto de estratégias de forma a permanecerem relevantes e resistirem ao ritmo da mudança".

(*Accenture*)

Resumo

O objetivo principal deste projeto de mestrado foi a criação de uma aplicação móvel na área da Banca que permita ao seu utilizador executar um conjunto de funcionalidades bancárias (como por exemplo consultar o estado atual da conta, efetuar pagamentos e realizar transações), sem ter de se deslocar fisicamente ao banco. O utilizador desta aplicação será assim o cliente do banco em questão.

O projeto desenvolveu-se na *Accenture*, uma empresa internacional com grande relevância no mercado português. Aqui foi elaborado um plano de estágio que incluiu análise e desenho da solução, desenvolvimento, testes e documentação. Inicialmente estava prevista uma colaboração com o banco, mas este cancelou algumas fases do projeto antes do seu término. Contudo, a *Accenture* continuou a suportar o projeto por ser um investimento na sua carteira de *software*.

O principal foco deste projeto foi a funcionalidade de transações, tanto entre contas do mesmo banco como entre contas de bancos diferentes. Foram tidas em consideração características de usabilidade, segurança e inovação para potenciar o interesse dos utilizadores na aplicação. Por exemplo, foi implementado um sistema de código de barras que permite gerar uma caixa digital para receção de uma futura transação, sendo que qualquer pessoa com a aplicação pode realizar a transferência simplesmente fotografando o código de barras; a usabilidade foi também considerada essencial, sendo que para a aumentar a solução foi prototipada antes da sua implementação em código; foram ainda seguidos princípios de segurança para evitar o uso malicioso da aplicação (autenticação baseada em palavra-chave ou em fatores biométricos, uso de métodos criptográficos para garantir a privacidade dos dados, etc.).

Palavras-chave: *Android*, aplicações móveis, dispositivos móveis, *mobile banking*, transações.

Abstract

The main objective of this master's project was to create a mobile application in the Banking area that allows its user to execute a set of banking functions (such as checking the current account status, making payments and making transactions), without having to physically move to the bank. The user of this application will thus be the client of the bank in question.

The project was developed in Accenture, an international company with great relevance in the Portuguese market. Here a training plan was elaborated that included analysis and design of the solution, development, tests and documentation. Initially a collaboration with the bank was planned, but this bank canceled some phases of the project before its end. However, Accenture continued to support the project as an investment in its software portfolio.

The focus of this project was the functionality of transactions, both between accounts of the same bank and between accounts of different banks. Usability, security and innovation characteristics have been considered to enhance the users' interest in the application. For example, a barcode system has been implemented that allows the generation of a digital box to receive a future transaction, and anyone with the application can perform the transfer simply by photographing the barcode; usability was also considered essential, and, to increase it, the solution was prototyped before its implementation in code; security principles were also followed to prevent malicious use of the application (keyword-based authentication or biometric factors, use of cryptographic methods to ensure data privacy, etc.).

Keywords: Android, mobile applications, mobile devices, mobile banking, transactions.

Conteúdo

Capítulo 1 Introdução 1

1.1 Motivação 1

1.2 Objetivos 3

1.3 Contribuições 3

1.4 Organização do documento 3

Capítulo 2 Enquadramento 5

2.1 Enquadramento da empresa *Accenture* 5

2.2 O contexto do projeto: o cliente 6

2.3 Calendarização do projeto 6

Capítulo 3 Revisão da literatura 8

3.1 Engenharia de *Software* 8

3.2 Metodologias 9

3.3 Segurança 13

3.4 Tecnologias 22

3.5 Rede de computadores 27

3.6 Aplicações móveis 28

3.7 A plataforma *Android* 30

Capítulo 4 Trabalho relacionado 38

4.1 Serviços bancários: enquadramento histórico 38

4.2 Evolução tecnológica associada ao sector bancário 39

4.3 Aplicação já existente 42

Capítulo 5 Trabalho desenvolvido 43

5.1 Descrição geral do projeto 43

5.2 Levantamento de requisitos 44

5.3 *Design* de *software* e base de dados 51

5.4 *Layouts* 62

5.5 Implementação 70

5.6 Testes 76

5.7 Rede 82

Capítulo 6 Conclusões..... 84

6.1 Considerações finais 84

6.2 Perspetivas futuras 85

Referências bibliográficas..... 87

Anexos…….. 90

Anexo A – Descrição dos casos de uso 90

Anexo B – Diagramas de sequência 114

Anexo C – Modelo de domínio..... 131

Anexo D – Contratos das operações 132

Anexo E – Diagramas de interação..... 144

Anexo F – Modelo de classes 155

Anexo G – Arquitetura lógica do sistema..... 157

Anexo H – Testes de sistema 160

Lista de Figuras

Figura 1.1: O cliente da banca *online*. 2

Figura 1.2: Utilização do serviço *Mobile Banking* através de *App* (%). 2

Figura 2.1: *Gantt chart* do projeto. 7

Figura 3.1: Camadas da Engenharia de *Software*. 9

Figura 3.2: Visão geral do processo de *Scrum*. 12

Figura 3.3: Visão geral do processo de RUP. 13

Figura 3.4: Processo de criptografia simétrica. 15

Figura 3.5: Processo de criptografia assimétrica. 16

Figura 3.6: Processo de autenticação e escolha do código pessoal. 18

Figura 3.7: Processo de autenticação para a realização dos diversos tipos de transferências. 19

Figura 3.8: Envio da informação entre cliente e servidor. 21

Figura 3.9: Exemplo de um código de barras PDF417. 26

Figura 3.10: Utilização de sistemas operativos (janeiro – novembro 2017). 30

Figura 3.11: Arquitetura da plataforma *Android*. 31

Figura 3.12: Dados relativos ao número de dispositivos de cada versão do *Android*. 32

Figura 3.13: Ciclo de vida de uma atividade. 35

Figura 4.1: O sistema bancário português (2010-2016). 39

Figura 4.2: Evolução do número de caixas automáticas (2011-2017). 40

Figura 5.1: Diagrama de controlo de fluxo de transferências entre contas do mesmo banco. 45

Figura 5.2: Diagrama de controlo de fluxo de transferências entre contas interbancárias. 45

Figura 5.3: Diagrama de casos de uso. 50

Figura 5.4: Arquitetura da base de dados. 58

Figura 5.5: Arquitetura em camadas do sistema.. 59

Figura 5.6: Arquitetura física do sistema. 60

Figura 5.7: Protótipos de baixa fidelidade. 64

Figura 5.8: Protótipos de alta fidelidade. 67

Figura 5.9: Diagrama de controlo de fluxo de transferências entre contas do mesmo banco. 68

Figura 5.10: Diagrama de controlo de fluxo de transferências entre contas interbancárias..... 68

Figura 5.11: Diagrama de controlo de fluxo de transferências por código (criação de código). 69

Figura 5.12: Diagrama de controlo de fluxo de transferências por código (leitura de código).. 69

Figura 5.13: Rede criada para efetuar testes de sistema. 82

Lista de Tabelas

Tabela 2.1: WBS (*Work Breakdown Structure*). 7

Tabela 3.1: Vantagens e desvantagens do modelo cascata *vs.* iterativo e incremental.. 10

Tabela 3.2: Descrição dos processos de autenticação. 17

Tabela 5.1: Requisitos funcionais. 47

Tabela 5.2: Requisitos não funcionais. 49

Tabela 5.3: Descrição dos casos de uso. 50

Tabela 5.4: Descrição dos diagramas de sequência. 51

Tabela 5.5: Descrição dos diagramas de interação. 53

Tabela 5.6: Descrição do modelo de classes da aplicação móvel. 55

Tabela 5.7: Descrição do modelo de classes dos serviços. 56

Tabela 5.8: Descrição da arquitetura da base de dados. 57

Tabela 5.9: Resultados dos testes realizados. 81

Tabela 5.10: Descrição dos elementos da rede. 83

Abreviaturas

ADT – *Android Development Tools*

AES – *Advanced Encryption Standard*

API – *Application Programming Interface*

ATM – *Automated Teller Machine*

AVD – *Android Virtual Devices*

CASE – *Computer Aided Software Enginnering*

DSDM – *Dynamic System Development Model*

FDD – *Feature Driven Development*

GSM – *Global System for Mobile Communications*

IBM – *International Business Machines Corporation*

IDE – *Integrated Development Environment*

IETF – *Internet Engineering Task Force*

IP – *Internet Protocol*

JVM – *Java Virtual Machine*

LTE – *Long Term Evolution*

MSF – *Microsoft Solutions Framework*

MVC – *Model-View-Controller*

NAT – *Network Address Translation*

PDA – *Personal Digital Assistants*

RAD – *Rapid Application Development*

REST – *Representational State Transfer*

RSA – *Rivest-Shamir-Adleman*

RUP – *Rational Unified Process*

SDK – *Software Development Kit*

SEPA – *Single Euro Payments Area*

SGBD – *DataBase Management System*

SHA – *Secure Hash Algorithm*

SIBD – Sistemas de Informação e Bases de Dados

SIBS – Sociedade Interbancária de Serviços

SMS – *Short Message Service*

SOAP – *Simple Object Access Protocol*

SQL – *Structured Query Language*

TEE – *Trusted Execution Environment*

UI – *User Interface*

UML – *Unified Modeling Language*

UMTS – *Universal Mobile Telecommunications System*

URL – *Uniform Resource Locator*

USB – *Universal Serial Bus*

USDP – *Unified Software Development Process*

WBS – *Work Breakdown Structure*

WCDMA – *Wide-Band Code-Division Multiple Access*

XP – *eXtreme Programming*

Capítulo 1

Introdução

1.1 Motivação

Após a conclusão da Licenciatura em Engenharia Informática da Faculdade de Ciências da Universidade de Lisboa (FCUL), decidi prosseguir os conhecimentos nesta área, com a inscrição no Mestrado em Engenharia Informática.

Depois do término da parte teórica do Mestrado (1º ano), resolvi elaborar o trabalho de projeto numa empresa externa, com grande relevância no mercado nacional, a *Accenture*. Esta escolha permitiu-me angariar e perspetivar a vida profissional futura.

A escolha do tema do trabalho de projeto "***Solução mobile – Banking, Transações***", deveu-se ao facto de atualmente as aplicações móveis fazerem parte do quotidiano da população e do seu potencial de evolução.

O ***mobile banking*** é definido como um produto ou serviço oferecido por um banco para realizar transações financeiras e não financeiras usando um dispositivo móvel, ou seja, um telemóvel, *smartphone* ou *tablet*. Atualmente a maioria das pessoas efetua grande parte das suas atividades pessoais, profissionais e sociais, através da utilização de dispositivos móveis (Elidol e Erol, 2014).

Nestas atividades encontram-se todas as tarefas relacionadas com as movimentações bancárias, de entre as quais transações/transferências e verificação do estado da conta bancária. Em Portugal, a taxa de dispositivos móveis com ligação à Internet é elevada, contudo a taxa de utilização do *mobile banking* ainda é reduzida comparativamente com outros países da União Europeia (Marktest, 2017).

Um estudo do *Basef Banca da Marktest* (2016) indica que 2 545 000 portugueses utilizam a *internet banking*, correspondente a 35% dos residentes no continente com idade igual ou superior a 15 anos e possuidores de conta bancária. Salienta-se ainda que os homens e a faixa etária entre os 25 – 34 anos são os que mais aderem a esta modalidade (Figura 1.1) (Morais, 2017).

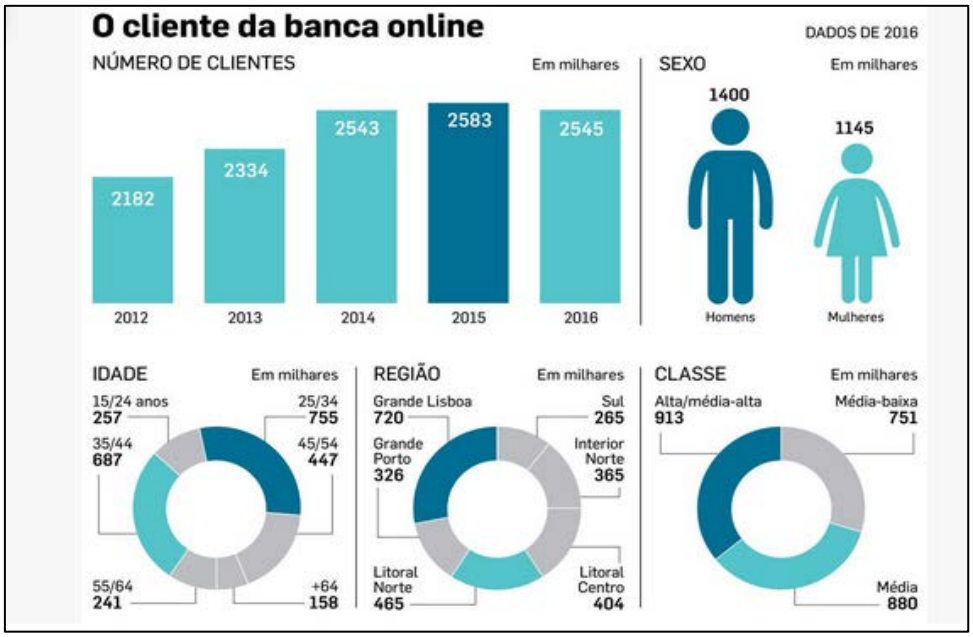


Figura 1.1: O cliente da banca *online*. Morais, 2017.

No que diz respeito à utilização de aplicações de *mobile banking*, o estudo indica que 650 000 portugueses recorrem a esta ferramenta (Marktest, 2017). Salienta-se também que a taxa de utilização de *mobile banking* entre os utilizadores do serviço de *internet banking* tem aumentado, passando de 18,3% em maio de 2014 para 25,1% no período homólogo de 2015 (Figura 1.2).

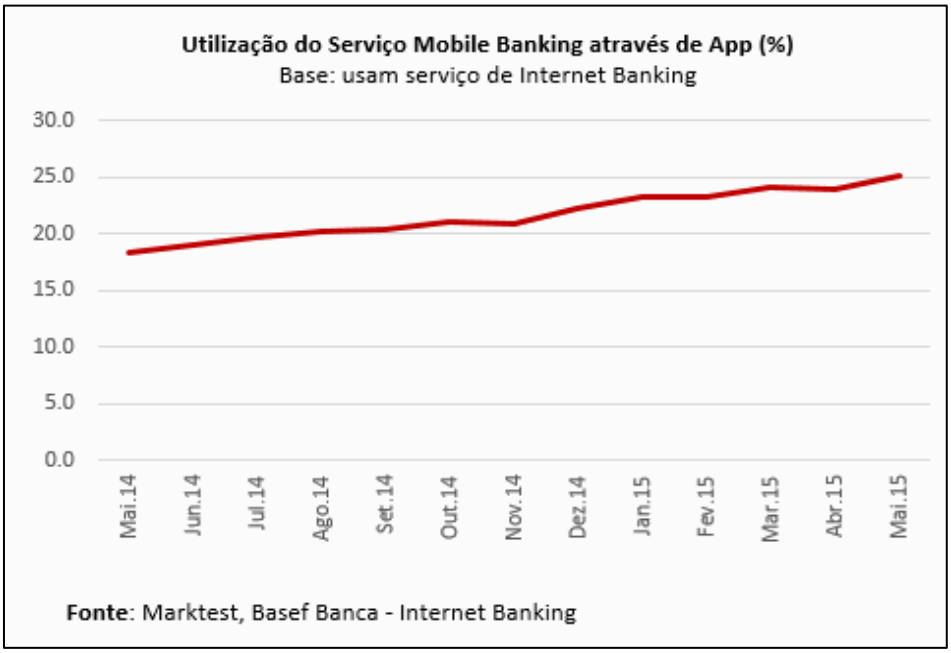


Figura 1.2: Utilização do serviço *Mobile Banking* através de *App* (%). Marktest, 2015.

Com o aumento da taxa de utilização de dispositivos e aplicações móveis por parte das instituições bancárias (ATM, *mobile* e *home banking*), surgiu a necessidade de facilitar as ações

dos consumidores finais. A solução passou pela criação de uma aplicação, em que o utilizador consegue aceder aos seus dados bancários e efetuar diversas funcionalidades (transações), através do seu telemóvel. Isto levou a que a maioria das instituições bancárias desenvolvessem aplicações para diversas plataformas (*Android*, iOS, *Windows*) no sentido de promover os seus produtos junto dos seus clientes, bem como facilitar o esforço desempenhado em tarefas bancárias.

1.2 Objetivos

O projeto de tese visou realizar uma aplicação para uma instituição bancária, onde está inserida uma funcionalidade que permite efetuar transações entre diferentes contas (contas do mesmo banco e contas interbancárias).

Para além do objetivo principal, este projeto pretendeu aplicar e aprofundar conhecimentos obtidos durante a Licenciatura e o primeiro ano do Mestrado em Engenharia Informática.

A *Accenture* tinha como finalidade a utilização de tecnologias e métodos inovadores para a realização de tarefas, de modo a melhorar a usabilidade de toda a aplicação *mobile*.

Este projeto permitiu também adquirir conhecimentos sobre a relação com o cliente, proporcionar e desenvolver competências técnicas, profissionais e sociais.

1.3 Contribuições

O projeto de tese contribuiu para a evolução tecnológica associada ao sector bancário português, tendo em consideração a história empresarial de Portugal, e o interesse que as aplicações móveis têm despertado neste sector e a sua utilização por parte dos clientes.

1.4 Organização do documento

O documento está estruturado do seguinte modo:

- **Capítulo 1** – introduz o aspeto motivacional, o tema sobre o qual incide o relatório de projeto e os objetivos delineados.
- **Capítulo 2** – incide sobre o enquadramento da empresa *Accenture*, o contexto do projeto e a sua calendarização.
- **Capítulo 3** – incide sobre o enquadramento do tema do projeto, com uma abordagem à Engenharia de *Software* e às metodologias, tecnologias e ferramentas empregues no projeto. Descrevem-se também os tipos de aplicações móveis existentes, bem como a plataforma *Android*.
- **Capítulo 4** – apresentação do trabalho relacionado. Enquadra o sector bancário português e a sua evolução tecnológica no relacionamento com o cliente. Engloba também o trabalho e as aplicações já existentes nesta área.

- **Capítulo 5** – incide sobre todo o trabalho desenvolvido durante o projeto, dividindo-se em subcapítulos (levantamento de requisitos, *design* de *software* e base de dados, *layouts* – desenho de interface, implementação, testes e ambiente de produção).
- **Capítulo 6** – conclui o documento, referindo se os objetivos delineados foram concretizados, os obstáculos durante a sua execução e a perspetiva futura.

Segue-se uma lista de referências bibliográficas e um conjunto de anexos que auxiliam a documentação de todo o projeto, em particular a parte de desenho de *software* e da implementação em código.

Capítulo 2

Enquadramento

2.1 Enquadramento da empresa *Accenture*

A *Accenture* é uma empresa global líder em serviços profissionais que oferece uma ampla gama de serviços e soluções em estratégia, consultoria, digital, tecnologia e operações.

Segundo um estudo do *Top Employers Institute* (2018), esta foi considerada a melhor empresa para trabalhar em Portugal, pela terceira vez consecutiva. De entre as razões apontadas para esta distinção, destacam-se:

- Compensação e benefícios;
- Condições de trabalho apropriadas;
- Desenvolvimento de liderança e cultura corporativa;
- Formação e desenvolvimento;
- Gestão de desempenho;
- Otimização das práticas no departamento de Recursos Humanos;
- Promoção do talento.

Os clientes da *Accenture* são provenientes de diversas indústrias de todo o mundo, permitindo que profissionais especializados possam desenvolver as competências valorizadas no mercado e ganhem experiência na sua atividade profissional. Ao nível dos serviços financeiros prestados destacam-se a Banca, o Mercado de Capitais, Seguros e Gestão de Riscos. Esta empresa trabalha com "*80% dos 50 maiores bancos do ranking Fortune Global 500*", disponibilizando estratégias para conquistar e fidelizar clientes, expandir a oferta de produtos e serviços, gerir o risco e explorar o potencial de novas tecnologias, com *softwares* melhorados, para garantir a satisfação dos clientes.

Num estudo da *Accenture* (As "*fintech*" e a evolução do mercado) são apresentadas algumas ideias para que o sector bancário resista à constante mudança e à evolução digital:

- Curto-prazo: melhorar as estratégias de negócio, recorrendo a tecnologias utilizadas noutros ramos da indústria;

- Médio-prazo: desenvolvimento de um programa de multi-investimento, centrado nos clientes e nos níveis de organização;
- Longo-prazo: recorrer à inovação e às plataformas tecnológicas para satisfazer os objetivos e necessidades dos seus clientes. Será necessário o recurso a grandes investimentos globais de longo prazo.

Dentro desta empresa, o projeto de tese a que este relatório se refere foi inserido num contexto da *Accenture digital* e *Accenture Technology*.

2.2 O contexto do projeto: o cliente

Como foi referido na secção 1.2, um dos principais objetivos deste projeto foi a realização de uma aplicação móvel para uma instituição bancária. No entanto, com o avançar do projeto, o cliente suspendeu determinadas fases do ciclo de desenvolvimento por motivos internos (ver Figura 2.1), levando a alterações no planeamento do projeto final.

Dado que a *Accenture* valoriza o desenvolvimento tecnológico, decidiu continuar a suportar o projeto de tese. Desta forma, proporcionou a realização de uma aplicação *mobile* para a Banca, utilizando novas tecnologias e métodos para efetuar operações bancárias, de modo a facilitar e melhorar a experiência de utilização, assegurando a eficiência e segurança.

O cancelamento do projeto por parte do cliente levou a alterações, em particular ao nível das infraestruturas: este projeto tinha um fim comercial (para uma empresa que opera no ramo bancário), e como tal, as infraestruturas usadas iriam pertencer ao cliente. Assim, as infraestruturas passaram a ser locais (na própria máquina). Tanto os serviços *web* (em *Java*) como a base de dados (SQL Server) foram implementadas localmente. Para usufruir dos componentes que estão na máquina local foi necessária a criação de uma sub-rede. Isto deveu-se ao facto de a máquina conter um IP (*Internet Protocol*) privado, quando se acede à Internet, não estando disponível para receber pedidos aos serviços. Sendo assim, através de uma sub-rede (endereços IP privados), todos os dispositivos ligados a essa rede podem comunicar entre si.

Para ultrapassar esta limitação, seria necessário transportar os serviços para um servidor com IP público ou comprar um IP e atribuir à máquina local, sendo possível aceder aos serviços através de qualquer cliente (IP) ligado a uma rede.

2.3 Calendarização do projeto

A calendarização do projeto foi estruturada com base no plano de trabalhos, facilitando o ciclo de vida do projeto, de modo a que os atrasos não afetassem a conclusão do mesmo. Assim sendo, a calendarização do projeto foi cumprida nas datas previstas, mesmo com o cancelamento pelo cliente. Na Tabela 2.1 e na Figura 2.1 estão representadas respetivamente a estrutura analítica e o *Gantt chart* do projeto.

Tabela 2.1: WBS (*Work Breakdown Structure*).

Nome da tarefa	Duração	Início	Fim
Análise e desenho da solução	42 dias	02-11-2017 (5ª feira)	29-12-2017 (6ª feira)
Levantamento de requisitos	10 dias	02-11-2017 (5ª feira)	15-11-2017 (4ª feira)
Análise funcional, técnica e arquitetural	16 dias	16-11-2017 (5ª feira)	07-12-2017 (5ª feira)
Desenho funcional	16 dias	08-12-2017 (6ª feira)	29-12-2017 (6ª feira)
Desenvolvimento de interfaces	23 dias	01-01-2018 (2ª feira)	31-01-2018 (4ª feira)
Esboços de baixa fidelidade	8 dias	01-01-2018 (2ª feira)	10-01-2018 (4ª feira)
Esboços de alta fidelidade	15 dias	11-01-2018 (5ª feira)	31-01-2018 (4ª feira)
Implementação, configuração e testes	86 dias	01-02-2018 (5ª feira)	31-05-2018 (5ª feira)
Implementação técnica da solução	60 dias	01-02-2018 (5ª feira)	25-04-2018 (4ª feira)
Configuração da solução integrada	16 dias	26-04-2018 (5ª feira)	17-05-2018 (5ª feira)
Realização de testes integrados	10 dias	18-05-2018 (6ª feira)	31-05-2018 (5ª feira)
Elaboração do relatório final	21 dias	01-06-2018 (6ª feira)	29-06-2018 (6ª feira)
Realização do relatório final da tese	21 dias	01-06-2018 (6ª feira)	29-06-2018 (6ª feira)

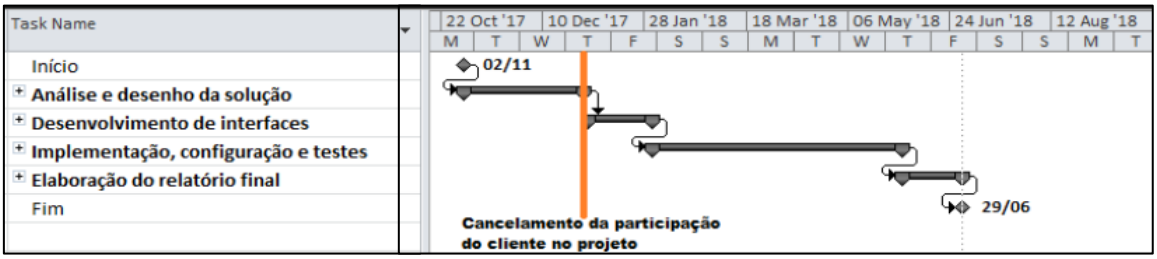


Figura 2.1: *Gantt chart* do projeto.

Capítulo 3

Revisão da literatura

3.1 Engenharia de *Software*

A engenharia deriva da palavra em latim *ingenium* e define-se como a formação técnica dos conhecimentos científicos com fim à criação, melhoramento e implementação de serviços funcionais para a vida do ser humano. Destacam-se vários tipos de engenharia como por exemplo: ambiental, biomédica, civil, computação, elétrica, genética, mecânica, de produção, química e de *software* (Pressman, 2005; Sommerville, 2011).

A **Engenharia de *Software*** é definida como a área do conhecimento da informática destinada à especificação, desenvolvimento e manutenção de sistemas de *software*. Foi na década de 50 do século XX, que surgiram os primeiros *softwares*, sendo que as pesquisas eram voltadas para o *hardware*, apenas disponível nos centros de pesquisa. Contudo, em 1960 foi utilizada pela primeira vez a designação de Engenharia de *Software*, pois o *software* tornou-se o grande foco dos cientistas e investigadores quando as dificuldades em desenvolver sistemas complexos foram discutidas (Pressman, 2005; Sommerville, 2011).

3.1.1 Estrutura e visão geral

A Engenharia de *Software* é composta por quatro camadas (Figura 3.1). O foco na qualidade é a parte fundamental, pois uma gestão da qualidade promove a melhoria e a eficácia dos processos e do produto final. Existe também uma camada de processos, sendo responsável pela coesão das camadas de métodos e ferramentas e permite o desenvolvimento do *software* dentro do prazo estabelecido. Além disso, define a sequência dos métodos que serão aplicados e as ferramentas que serão disponibilizadas. Seguidamente, a camada de métodos é responsável pelas informações técnicas para desenvolver o *software*. Existem diferentes métodos para as diferentes fases de desenvolvimento e para análise de requisitos, projeto, codificação, testes e manutenção. Por fim, a camada das ferramentas é responsável pela automatização do suporte para o processo e os métodos. Assim sendo, distinguem-se: CASE (*Computer Aided Software Enginnering*), análise estruturada, análise essencial ou orientada para objetos e linguagens de programação (Pressman, 2005).

Figura 3.1: Camadas da Engenharia de *Software*. Pressman, 2005.

De acordo com Pressman (2005), a estrutura de um processo aplicável à maioria dos projetos de *software*, independentemente do seu grau de complexidade, inclui os seguintes tópicos:

- ***Communication*:** é responsável pela comunicação e colaboração com o cliente, fazendo-se o levantamento de requisitos.
- ***Planning*:** é responsável pela descrição das tarefas técnicas, os riscos associados, os recursos necessários, os produtos finais e um cronograma.
- ***Modeling*:** é responsável pela construção de modelos para uma melhor compreensão dos requisitos do *software*.
- ***Construction*:** é responsável pela criação dos códigos e pela realização de testes.
- ***Deployment*:** é responsável pela avaliação e *feedback* do cliente quanto ao *software* desenvolvido.

Apesar da existência de processos de *software* distintos, definem-se quatro atividades comuns entre todos os processos (Sommerville, 2011):

- especificação de *software*;
- projeto e implementação de *software*;
- validação de *software*;
- evolução de *software*.

Desta forma, pode concluir-se que os principais atributos de um bom *software* são: facilidade de manutenção, eficiência, usabilidade, confiança e segurança. Dado que o projeto se enquadrou num desenvolvimento para aplicações móveis, este utilizou as ferramentas, os métodos e os processos empregues na Engenharia de *Software*.

3.2 Metodologias

Com a evolução dos dispositivos móveis, aumentam também as necessidades de criar, gerir e manter *software* para as aplicações móveis. Torna-se importante selecionar uma metodologia de projeto, que reúna as melhores soluções e vantagens para o cliente e empresa (Elidol e Erol, 2014).

Todos os projetos têm um conjunto de riscos associados e quanto mais cedo forem evitados no seu ciclo de vida, mais cedo o produto é disponibilizado para o consumidor final.

Isto leva a que o planeamento e a metodologia empregue se tornem importantes e tenham uma relevância extra no projeto. Segundo Roger S. Pressman (2005), na Engenharia de *Software* podem definir-se os seguintes modelos de processos: modelo em cascata; modelos de processos evolutivos (modelo baseado em protótipos e modelo em espiral); modelos de processos incrementais (modelo incremental e modelo RAD – *Rapid Application Development*); modelo unificado (USDP) e modelo *agile*. Dado que o modelo em cascata já se encontra em desuso no desenvolvimento de projetos, pois não suporta iterações no ciclo de desenvolvimento e não permite atualizações rápidas, neste projeto utilizou-se um **modelo iterativo e incremental**. A razão da sua escolha deveu-se ao facto de possibilitar as alterações e as atualizações que poderiam ocorrer nos objetivos delineados inicialmente pela *Accenture*. Os padrões mais conhecidos que seguem o modelo iterativo e incremental de desenvolvimento são o **RUP** (***Rational Unified Process***) e os modelos de desenvolvimento *agile* (Larman e Basili, 2003).

De acordo com Roger S. Pressman (2005), enumeram-se na Tabela 3.1 as principais vantagens e desvantagens do modelo cascata e do modelo iterativo e incremental.

Tabela 3.1: Vantagens e desvantagens do modelo cascata *vs.* modelo iterativo e incremental.
Adaptado de: Pressman, 2005.

Modelo cascata	Modelo iterativo e incremental
Vantagens	
Utilizado em pequenos projetos.	Maior liberdade no planeamento do projeto e em cada etapa do trabalho.
Produtos muito similares construídos pela mesma equipa.	Equipa de trabalho mais unida e a divisão do trabalho é realizada de acordo com as competências de cada elemento.
Requisitos bem definidos e pouca expectativa de mudança.	Participação mais ativa do cliente em todas as fases do projeto, através de *feedbacks*.
Desvantagens	
Dificuldade do cliente em especificar todos os requisitos.	Produto entregue por partes, perdendo importância caso o cliente necessite de um produto 100% pronto.
O cliente tem de ser paciente.	Planeamento extenso, exigindo várias análises em cada fase do projeto.
Os projetos reais raramente seguem o fluxo sequencial proposto.	Como o planeamento pode ser dividido por etapas, o custo poderá também sofrer aumentos.

3.2.1 Metodologia *agile*

A metodologia *agile* corresponde a um conjunto de práticas utilizadas durante o desenvolvimento de *software*. Atualmente, a maioria dos projetos são geridos por metodologias *agile*, devendo-se ao facto de possibilitar uma maior participação por parte da equipa, redução do tempo de introdução de um projeto no mercado, aumento da produtividade e da capacidade de gerir atualizações na evolução do projeto. Os principais tipos desta metodologia são: FDD

(*Feature Driven Development*), XP (*eXtreme Programming*), MSF (*Microsoft Solutions Framework*), DSDM (*Dynamic System Development Model*) e *Scrum* (Elidol e Erol, 2014).

3.2.1.1 *Scrum*

Atualmente, a *Accenture*, tal como a maioria das empresas, utiliza a metodologia ***Scrum*** nos seus projetos. O *Scrum* é uma estrutura metodológica usada para implementar a metodologia *agile*, sendo o mais utilizado hoje em dia. A sua aplicabilidade não se estende apenas ao desenvolvimento de *software* (Elidol e Erol, 2014).

Este método divide o projeto em pequenos períodos, chamados *sprints*, em que cada um corresponde a um mini projeto, contento as habituais fases de um projeto – planeamento, análise de requisitos, *design* de *software*, desenvolvimento, testes e documentação. Estes são ciclos de desenvolvimento que começam numa reunião de planeamento (*Sprint Planning*) e terminam ou na demonstração do produto (*Sprint Review*) ou na revisão de processos utilizados no ciclo (*Sprint Retrospective*) (Santos *et al*., 2016).

De acordo com Ken Schwaber (2004) na metodologia *Scrum* existem três funções definidas para a divisão dos membros da equipa envolvidos num projeto:

- *Product Owner*: é responsável por definir quais as prioridades a serem desenvolvidas em cada *sprint*, sendo a ponte de ligação entre a área de negócio e a equipa de *Scrum*.
- *Scrum Master*: é responsável pela equipa, garantindo que esta siga a metodologia *Scrum*, não permitindo interrupções externas. Ajuda a equipa a realizar as suas tarefas com a melhor eficácia possível.
- *Scrum Team*: corresponde à equipa de desenvolvimento. Todos devem cumprir com os prazos estipulados em cada *sprint*. Deve ser uma equipa multidisciplinar e que não envolva muitos participantes.

Os ciclos encontram-se divididos em fases de modo a estruturar o trabalho realizado num *sprint* (Figura 3.2). Deste modo existem quatro fases durante uma iteração (Schwaber, 2004):

- *Sprint Planning*: o planeamento do *sprint* é efetuado nessa reunião, onde é definido como será realizado o trabalho da equipa de acordo com o período estabelecido. Todas as funcionalidades pretendidas devem ficar descritas (*Product Backlog*) que serão integradas no *Sprint Backlog*.
- *Daily Scrum*: corresponde a uma reunião diária de curta duração. Permite que a equipa discuta os prós e contras do que tem sido efetuado durante a realização do *sprint*.

- *Sprint Review*: trata-se de uma reunião realizada no final de cada *sprint* para discussão do que foi desenvolvido no ciclo. Cabe ao *Product Owner* a decisão, se a tarefa foi concluída ou se deverá ser incluída num novo *sprint*.
- *Sprint Retrospective*: esta reunião acontece entre a *Sprint Review* e a *Sprint Planning*. Permite detetar desvios que possam ocorrer e aperfeiçoar aquilo que já está delimitado.

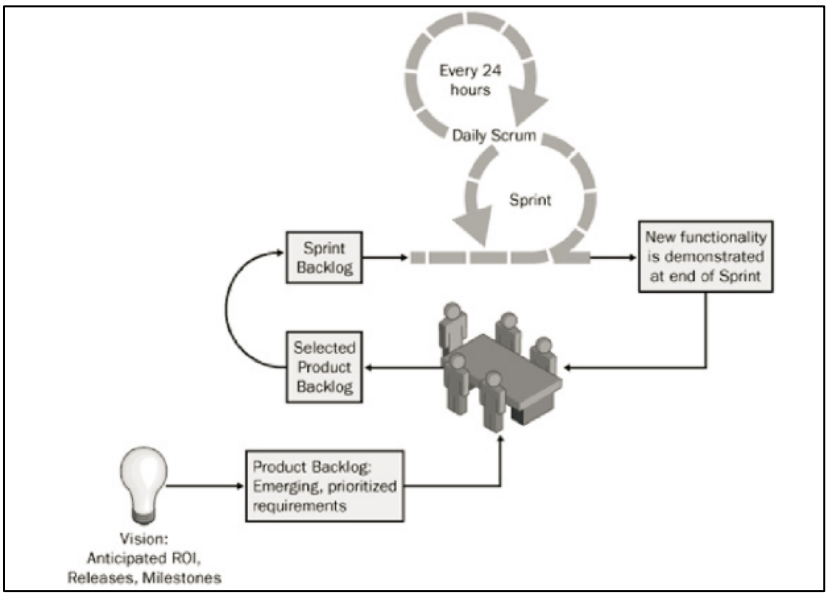


Figura 3.2: Visão geral do processo de *Scrum*. Schwaber, 2004.

3.2.2 RUP

Como o projeto se tornou interno, não existiu a necessidade de se utilizar a metodologia *agile*. Portanto, houve a necessidade de encontrar outro método de desenvolvimento para facilitar a concretização do projeto em causa. Visto que o modelo em cascata (modelo tradicional) torna o desenvolvimento bastante limitado em atualizações, e como a utilização da metodologia *agile* se tornou desnecessária, adotou-se o **RUP**.

O RUP corresponde a um processo da Engenharia de *Software* baseado no desenvolvimento iterativo e incremental, facilitando atualizações e alterações que venham a ocorrer no ciclo de vida do projeto. Foi criado pela *Rational Software Corporation*, sendo adquirido em 2003 pela IBM (*International Business Machines Corporation*). É bastante utilizado em áreas tais como: telecomunicações, transportes e serviços financeiros (Kruchten, 2004).

Um projeto que usa o RUP tem um ciclo de vida que consiste em várias iterações, conforme é visível na Figura 3.3. Uma iteração sustenta um determinado conjunto de tarefas quase sequencial (modelo de negócio, análise de requisitos, *design*, implementação, testes e

passagem a produção) em várias proporções, dependendo da zona onde se encontra o ciclo de desenvolvimento (Kruchten, 2004).

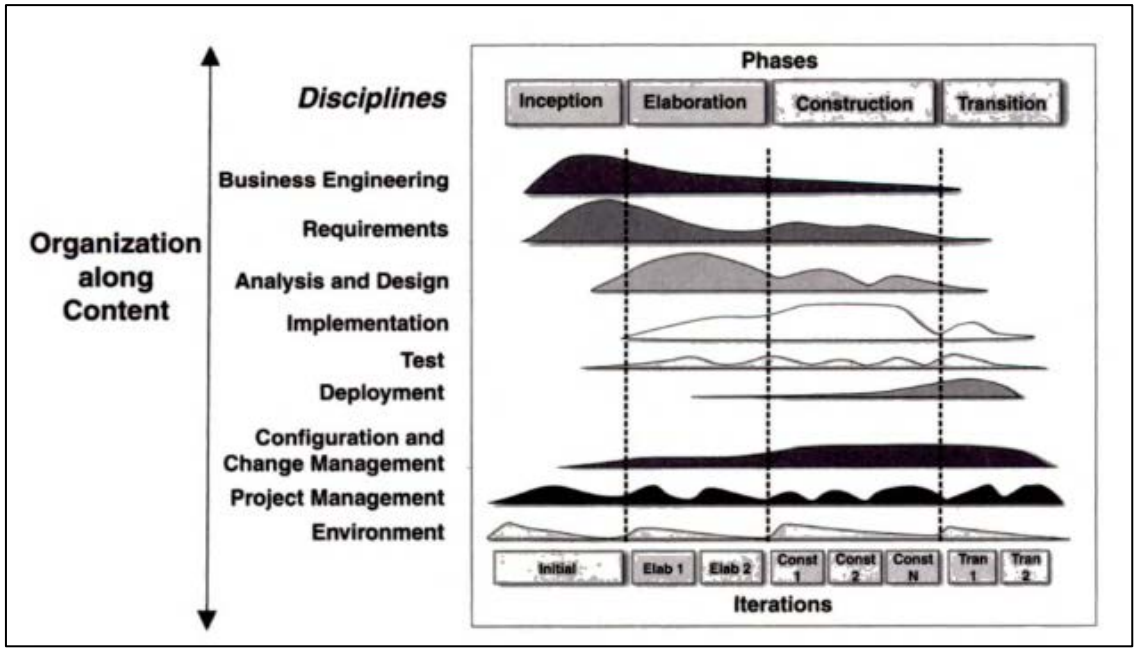


Figura 3.3: Visão geral do processo de RUP. Kruchten, 2004.

Segundo Philippe Kruchten (2004), a organização do desenvolvimento de *software* é efetuada em quatro fases: fase de iniciação, fase de elaboração, fase de construção e fase de transição. Neste projeto, as iterações iniciais (iniciação e elaboração) concentraram-se nas atividades *modeling* de negócio, análise de requisitos e *design*, não existindo praticamente o processo de implementação. As iterações na fase de construção reuniram o *design*, a implementação e os testes. Foi nesta fase que ocorreu a maior parte da codificação. As iterações na fase de transição têm como principal objetivo assegurar que o *software* seja disponibilizado para o cliente final, através da execução de diversos tipos de testes. Neste projeto apenas foi abordado o levantamento de requisitos, *design*, implementação e testes.

3.3 Segurança

Como já foi referido, as aplicações fazem parte da rotina dos utilizadores e estas fazem recurso de muitos dados privados dos mesmos. Tornam-se essenciais as questões relativas à segurança e privacidade dos dados nas aplicações. A segurança numa aplicação móvel não se destina apenas ao conjunto de permissões facultadas às aplicações para acederem aos seus dados privados. Engloba também o modo como determinados dados são armazenados e transferidos entre os diversos componentes num sistema aplicacional (por exemplo: comunicação entre aplicação móvel e serviços).

Devido à enorme procura, o mercado *Android* tornou-se muito competitivo, fazendo com que as empresas queiram lançar os seus produtos o mais rapidamente possível, atrasando, para futuras atualizações, a segurança das aplicações.

De acordo com um estudo da Universidade de Valladolid (2014), quando é desenvolvida uma aplicação *mobile*, é necessária a implementação dos seguintes tópicos:

- Alertas de falhas de segurança: necessitam de possuir um sistema de deteção de falhas de segurança, para o caso de alguém conseguir ter acesso indevido aos dados de um determinado utilizador.
- Autenticação: deve conter um mecanismo de autenticação para poder aceder a dados ou efetuar funcionalidades relevantes. Esta autenticação deve ser efetuada através de um ID único, senha, *pin* ou fatores biométricos (impressão digital).
- Controlo de acesso: uma aplicação deve limitar o acesso de dados privados a outros utilizadores da aplicação. Deve ser uma aplicação orientada para o utilizador.
- Informação dos utilizadores: todas as aplicações devem praticar uma política de privacidade de dados, encriptando-os antes de os armazenar no sistema. O sistema não deve operar com as senhas originais dos utilizadores, mas com um *hash* da mesma, permitindo assim que a segurança melhor consideravelmente.
- Transferência de dados: os dados enviados pela rede devem passar previamente por métodos de criptografia.

3.3.1 Funções de *hash*

Uma função de *hash* permite processar uma mensagem original de tamanho variável num valor de tamanho fixo. Após o término do algoritmo, torna-se praticamente impossível reverter o processo, ou seja, obter a mensagem original apenas com o valor de saída. Este tipo de funções deve possuir as seguintes características:

- Facilidade em obter o valor de saída para qualquer mensagem (rapidez no processo);
- Dificuldade em obter a mensagem original após passagem pelo algoritmo;
- Dificuldade em modificar a mensagem, sem modificar o seu *hash*;
- Duas mensagens originais diferentes não devem produzir a mesma saída.

As funções mais utilizadas na atualidade são da família de funções de dispersão SHA (*Secure Hash Algorithm*) (Gueron e Krasnov, 2012).

3.3.2 Criptografia simétrica e assimétrica

A criptografia foi inicialmente associada a atividades militares e diplomáticas. Tem como principal objetivo: ocultar informação através de processos de codificação, bem como repor essa mesma informação no seu estado original através de processos de descodificação (Pinto, 2010).

Em criptografia, a encriptação é o processo de transformação de uma informação original, numa informação ilegível para terceiros. Este mecanismo tem como objetivo o envio de informação confidencial de forma segura, sendo apenas possível a sua descodificação por pessoas autorizadas. Por outro lado, a desencriptação efetua a operação inversa da encriptação, transformando a informação ilegível no texto original através da chave de desencriptação (Pinto, 2010).

3.3.2.1 Criptografia simétrica

São algoritmos para criptografia que usam uma única chave para as funções de encriptação (cifrar) e desencriptação (decifrar) (Figura 3.4). A criptografia simétrica é também conhecida por criptografia de chave secreta, existindo diversos algoritmos de criptografia simétrica (por exemplo: DES, AES e RC4). Atualmente já são conhecidos ataques aos algoritmos, sendo necessário verificar qual o melhor algoritmo a ser utilizado num determinado contexto. Em comparação com os algoritmos da criptografia assimétrica, são mais rápidos, mas menos seguros, pelo facto de a chave ser partilhada entre os diversos componentes na comunicação (Pinto, 2010).

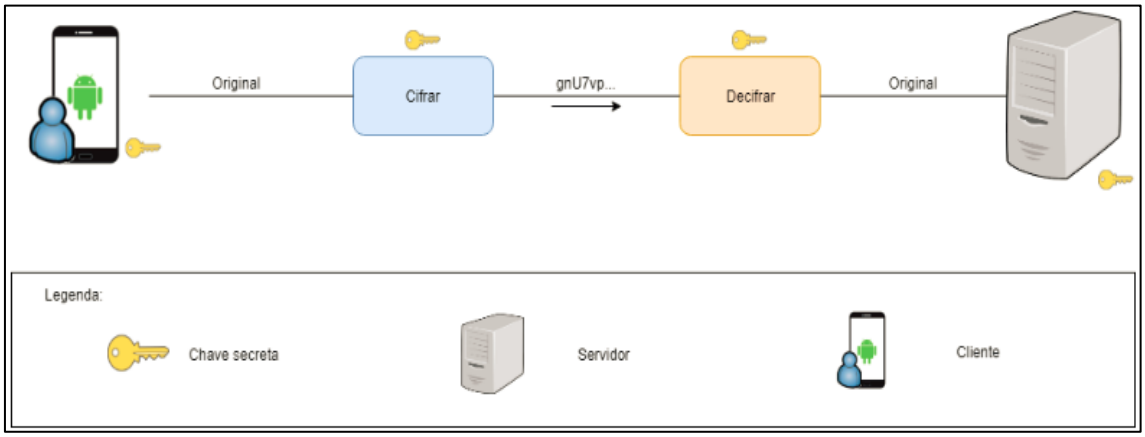


Figura 3.4: Processo de criptografia simétrica. Adaptado de: Pinto, 2010.

3.3.2.2 Criptografia assimétrica

É também designada por criptografia de chave pública, sendo uma classe de protocolos de criptografia baseados em algoritmos que necessitam de duas chaves (uma privada e uma pública). Apesar de serem diferentes, possuem ligações matemáticas que se revelam nas operações de encriptação (cifrar) e desencriptação (decifrar) (Figura 3.5).

A chave pública é distribuída pela rede onde todos os clientes têm acesso à mesma, dando a permissão de encriptar o texto original ou efetuar a verificação de uma assinatura digital. Por outro lado, a chave privada é reservada a um agente e não difundida na rede. Possibilita a desencriptação da informação previamente criptografada ou cria uma assinatura digital a ser verificada posteriormente. Existem diversos algoritmos que fazem uso da criptografia assimétrica, sendo o RSA (*Rivest-Shamir-Adleman*) dos mais utilizados (Pinto, 2010).

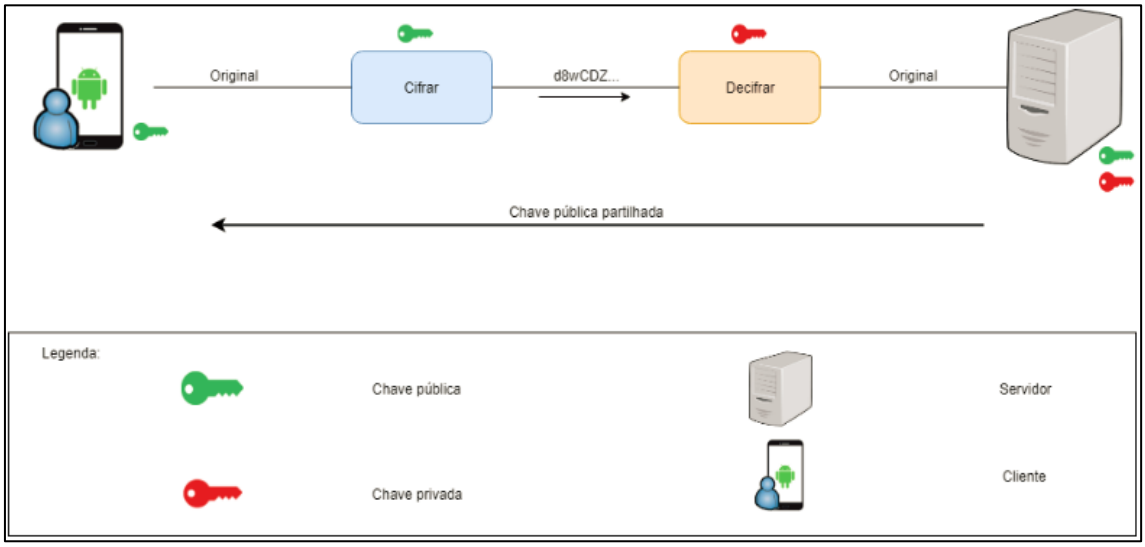


Figura 3.5: Processo de criptografia assimétrica. Adaptado de: Pinto, 2010.

3.3.3 Segurança no projeto

Durante todo o projeto existiu uma preocupação com a segurança e o modo como os dados eram enviados para a rede, pois como foi uma aplicação associada ao sector bancário, houve a necessidade de ser operar com os dados financeiros dos utilizadores.

A segurança neste projeto foi dividida em quatro partes:

1) Passwords de utilizadores;
2) Processo de autenticação;
3) Envio de dados pela rede;
4) Armazenamento de dados no servidor.

3.3.3.1 Passwords de utilizadores

As passwords de utilizadores desde sempre que se tornaram muito importantes para os sistemas informáticos, mas ainda continuam a ser desvalorizadas por grande parte dos utilizadores. São um dos principais acessos às aplicações, que permitem garantir confidencialidade e autenticidade, e por isso contêm grande relevância na área da informática.

No projeto, com o objetivo de as manter com um maior nível de segurança, o sistema passou a password juntamente com um *salt* por uma função de *hash*, de modo a trabalhar com o *hash* e não com a password original. O algoritmo usado para obter o *hash* da password foi o SHA-256.

O SHA-256 é a versão mais conhecida da família SHA-2. O valor 256 representa o número de *bits* que são originados (de saída) pelo algoritmo. É um algoritmo muito referido em estudos científicos de segurança informática, estando ligado a diversos processos no *software Debian GNU/Linux* e existindo movimentações de fornecedores da *Unix* e *Linux* para a utilização deste algoritmo, bem como do SHA-512. A estrutura do 512 é igual à do 256 e as diferenças encontram-se no número de ciclos que cada um executa e no número de *bits* de saída (256 *bits* e 512 *bits* respetivamente) (Gueron e Krasnov, 2012).

3.3.3.2 Processo de autenticação

Nas aplicações que trabalham com dados relevantes do público-alvo, o processo de login torna-se insuficiente para garantir a autenticidade de um indivíduo. Este fator deve-se aos inúmeros ataques aos dados financeiros dos membros de uma instituição bancária. Assim, para aumentar a proteção de dados foi implementado um sistema de autenticação antes de o utilizador terminar o login bem como antes de realizar qualquer funcionalidade relevante, por exemplo, transferências. Existem duas possibilidades para a autenticação: através de um código pessoal ou por meio de fatores biométricos (impressão digital) de modo a facilitar a utilização da aplicação.

As duas figuras abaixo representam os processos de autenticação utilizados no projeto, com recurso à ferramenta *draw.io*. A Figura 3.6 representa a lógica que está no caso de uso do login do utilizador, enquanto a Figura 3.7 representa o esquema lógico da autenticação nos processos de realização de transferências. Na Tabela 3.2 descrevem-se detalhadamente os processos de autenticação acima referidos.

Tabela 3.2: Descrição dos processos de autenticação.

Sequência	Descrição
1	O sistema verifica se o utilizador já possui código pessoal. Ocorre no primeiro acesso à aplicação e até o utilizador não escolher um código. A aplicação encaminha o utilizador para a introdução do seu número de telemóvel.
2	O sistema confirma que o utilizador já se encontra com um código pessoal. Dirige-o para o processo de autenticação (código pessoal ou impressão digital).
3	O sistema detetou alguma irregularidade. O número de telemóvel não corresponde ao do utilizador em curso de login ou o código SMS digitado não é igual ao que foi enviado.
4	Neste passo o utilizador já contém o seu código pessoal. Tanto o código SMS como o código pessoal cumpriram com os requisitos de segurança.

Sequência	Descrição
5	A autenticação ou a funcionalidade foi realizada com sucesso. A impressão digital ou as quatro posições inseridas pelo utilizador do seu código pessoal, como o ID do dispositivo estão de acordo com os dados do utilizador.
6	Autenticação duvidosa – As quatro posições do código pessoal ou a impressão digital detetada estão corretas, mas existiu uma falha no ID do dispositivo. Pode pensar-se que alguém está a tentar aceder indevidamente aos dados de um utilizador. É criado um código que, será enviado para o email do utilizador e, só terá sucesso na autenticação quando digitar corretamente o código enviado.
7	Erro ao inserir as quatro posições ou a detetar a impressão digital ou na introdução do código na autenticação duvidosa.

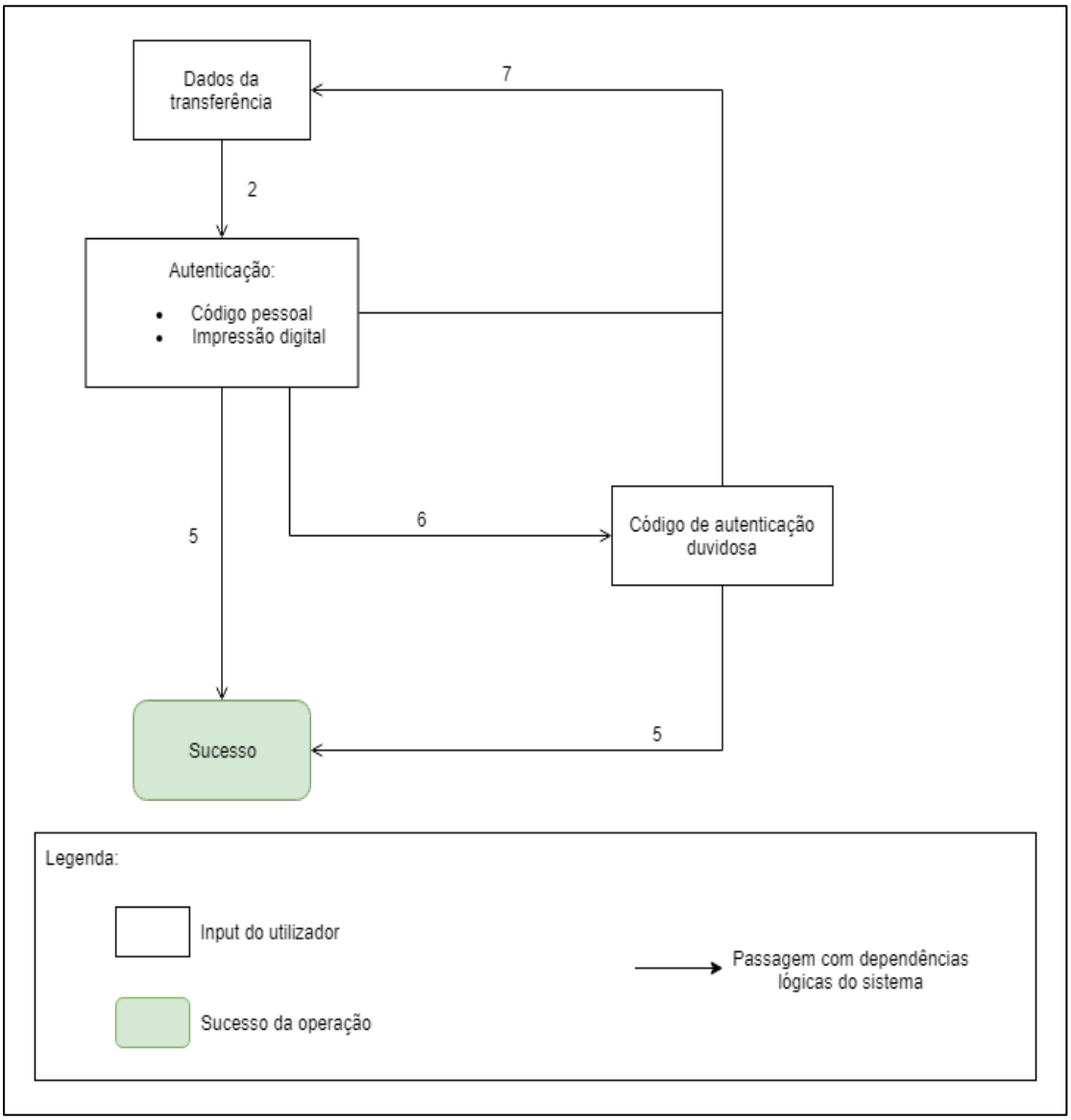


Figura 3.6: Processo de autenticação e escolha do código pessoal.

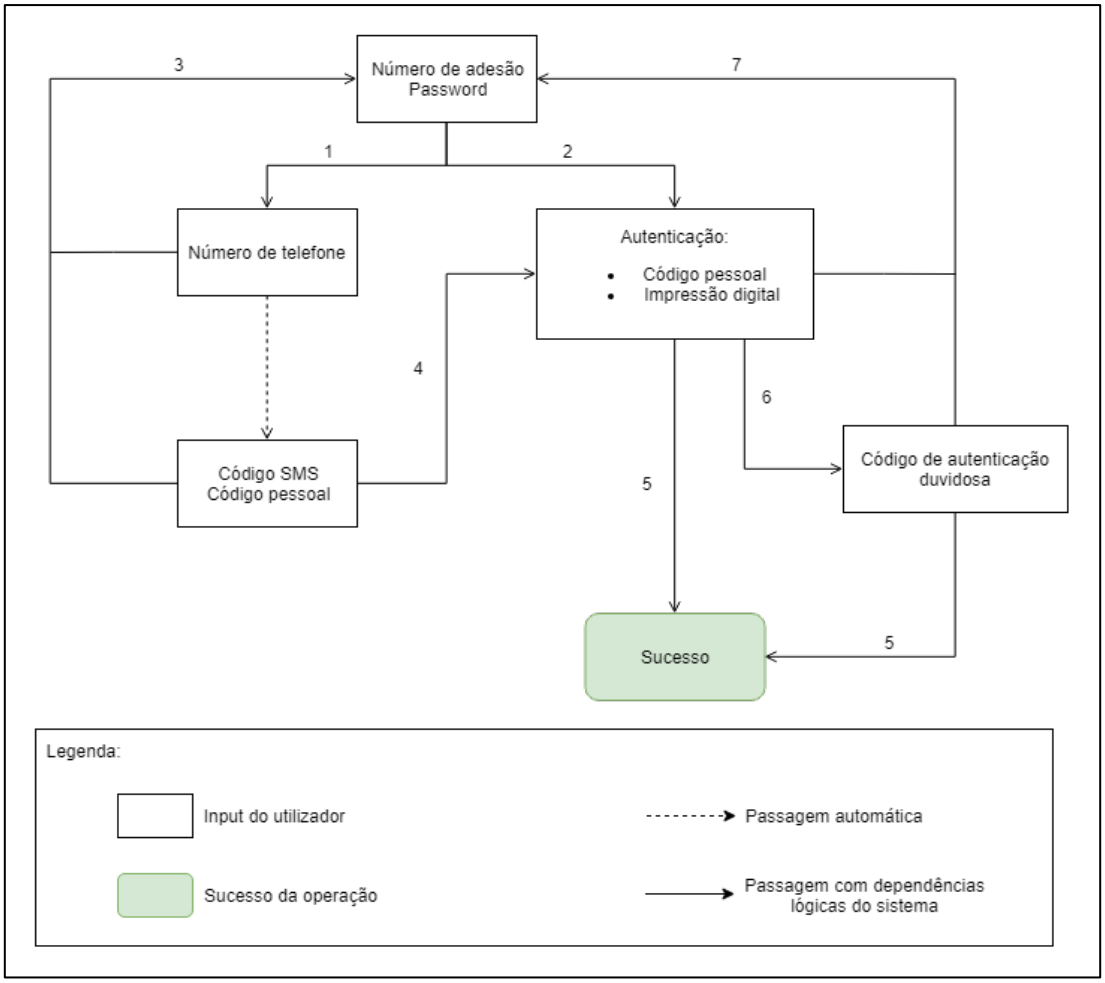


Figura 3.7: Processo de autenticação para a realização dos diversos tipos de transferências.

Uma das intenções do processo de autenticação era agilizar esta fase, de modo a diminuir o esforço por parte do utilizador para realizar tarefas na aplicação, mantendo os níveis de segurança. Foi incluída uma autenticação utilizando fatores biométricos (impressão digital), melhorando a usabilidade. Possui também uma autenticação através de quatro posições aleatórias de um código pessoal escolhido pelo utilizador. Existiam outras tecnologias e métodos de autenticação que melhoravam o sucesso das funcionalidades (usabilidade), mas diminuíam a segurança (padrão de desbloqueio ou introdução do código pessoal por completo, em vez das quatro posições).

Padrão de desbloqueio *vs.* Código

O padrão de desbloqueio é um desenho que o utilizador efetua no seu dispositivo entre nove pontos distantes desenvolvido pela *Google*. O padrão tem diversas características de desenho e é reconhecido como um código perante o sistema. Cerca de 40% dos utilizadores de *Android* utilizam este método de autenticação para guardar os seus dados. Contudo já foram encontradas diversas lacunas neste processo. Ye *et al.* (2017), desenvolveram um algoritmo que descobriu o padrão de desenho, observando o movimento dos dedos/mãos a uma distância e

ângulos variáveis de 2 a 9 metros. O estudo contou com 215 utilizadores e 120 padrões únicos, sendo que 95% dos padrões analisados foram descobertos pelo algoritmo em cinco ou menos tentativas. Referiram ainda que, este método de autenticação é limitado em segurança e que os códigos mais simples eram mais dificilmente descodificados do que os mais complexos. Este facto está relacionado com a precisão do movimento dos dedos, isto é, quanto mais complexo o padrão, maior é a precisão no movimento e é mais facilmente reconhecido pelo algoritmo.

Inserção de quatro posições *vs.* Código completo

A utilização do código completo contempla os mesmos problemas de segurança descritos no estudo de Ye *et al*. (2017). Isto é, se o utilizador introduzir o código completo poderia ser decifrado o movimento dos dedos e estaria comprometida a segurança da conta bancária. No entanto, como o utilizador introduz apenas quatro posições do código geradas aleatoriamente, apenas poderão ser decifradas essas posições, dificultando o processo de autenticação futuro. Sempre que existe uma autenticação, a aplicação irá pedir quatro posições novas, fazendo com que os processos de autenticação sejam independentes uns dos outros. Este processo diminui a usabilidade, pois requer esforço para saber qual o dígito na posição solicitada, mas aumenta a segurança. Contudo trata-se de um código digitado pelo utilizador, logo de fácil memorização para o mesmo.

Evidencia-se também que este processo é vulgarmente utilizado no sector da Banca em Portugal, para os clientes realizarem operações na *internet* e *mobile banking*.

3.3.3.3 Envio de dados pela rede

Com a evolução da tecnologia e o aumento da utilização de aplicações existiu a necessidade de criar um modelo que permitisse ligar vários clientes a uma máquina (modelo cliente-servidor). Este modelo corresponde a uma arquitetura na qual o processamento da informação é dividido em processos distintos. Existem processos responsáveis pela manutenção da informação e processamento lógico (servidores) e outros responsáveis pela apresentação e obtenção de dados (clientes). Os clientes efetuam pedidos aos servidores e estes processam e enviam os resultados. Devido à grande capacidade de processamento, os servidores suportam os componentes mais pesados na rede, deixando apenas a apresentação de dados para as máquinas dos clientes (Oluwatosin, 2014).

A aplicação seguiu o modelo cliente-servidor, uma vez que será usada por um grande número de utilizadores e existirá a necessidade de difundir dados na rede. Com o objetivo de proteger esses dados na rede foi essencial criar uma política para enviar os dados até ao servidor. Essa política englobou a utilização de duas técnicas de criptografia (simétrica e assimétrica), já descritas anteriormente. Na criptografia simétrica foi utilizado o algoritmo AES (*Advanced Encryption Standard*), onde o seu principal objetivo foi encriptar os dados a serem enviados. Por outro lado, na criptografia assimétrica há a utilização do algoritmo RSA, onde o

seu principal objetivo é a encriptação da chave utilizada anteriormente através da chave pública, para poder ser enviada para o servidor. No projeto utilizou-se a cifração com AES, porque é muito mais rápido do que com RSA, e se houver necessidade de cifrar várias mensagens, este esquema aumenta a velocidade de processamento.

A Figura 3.8 é uma representação lógica de como a aplicação (cliente) envia dados para o servidor, com a seguinte sequência de operações:

1) O cliente faz um pedido ao servidor para obter a sua chave pública;
2) O cliente pretende enviar uma informação para o servidor (por exemplo: a palavra "Original");
3) O cliente cria uma chave secreta (AES) e encripta os dados com essa chave. Sempre que pretende fazer um pedido ao servidor, vai criar uma chave nova;
4) O cliente encripta a chave secreta (AES) com a chave pública do servidor;
5) O cliente envia a chave e a informação encriptadas para o servidor;
6) O servidor vai obter a chave secreta utilizando o processo de desencriptação através da chave privada. Apenas consegue desencriptar se o par chave pública/privada for o mesmo;
7) O servidor vai obter a informação, fazendo uso da função de desencriptação através da chave secreta anteriormente decifrada. O servidor efetua o processamento, encripta a resposta com a chave secreta e envia-a para o cliente.

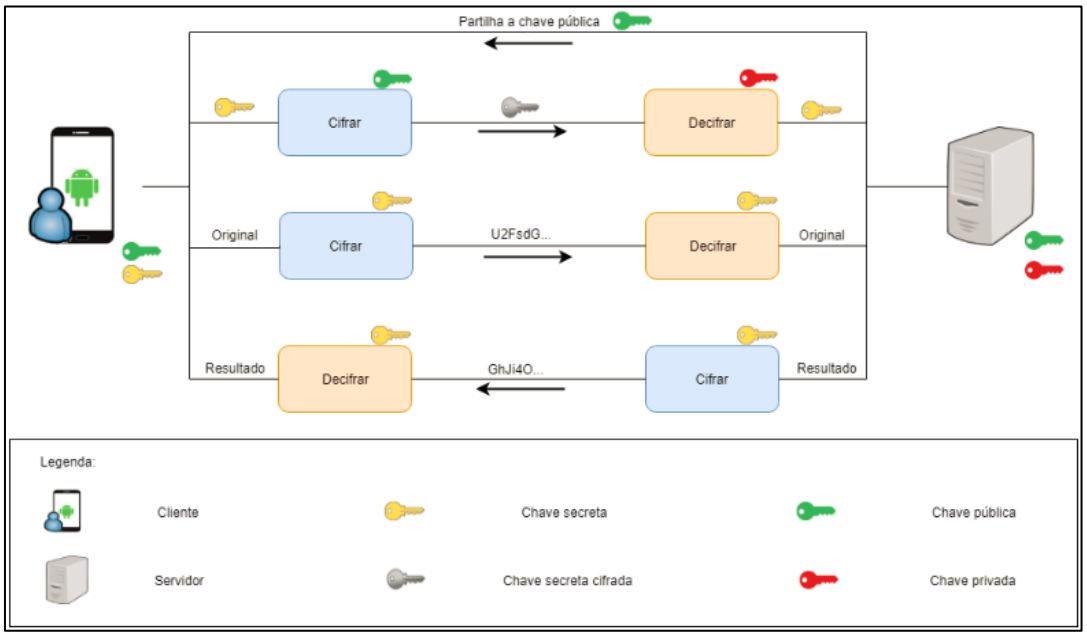


Figura 3.8: Envio da informação entre cliente e servidor.

3.3.3.4 Armazenamento de dados no servidor

Os dados são muito importantes para as aplicações e desse modo, devem ser guardados e preservados o mais cifrados possível para prevenir ataques de *sniffing*. Os ataques *sniffing* observam e analisam os dados de um computador ou dispositivo móvel, numa rede ou base de dados, de modo a obterem dados de utilizadores, senhas, nomes, códigos ou comportamentos, muitas vezes com um fim malicioso (Pinto, 2010).

Sempre que seja necessário armazenar alguma informação na base de dados, o servidor vai encriptar a informação com a sua chave pública. Quando necessitar de efetuar uma resposta a um pedido, vai decifrar a informação através da sua chave privada. Deste modo, os dados são armazenados de forma segura. No entanto, este processo aumenta o tempo de resposta dos serviços.

3.4 Tecnologias

Nesta secção vão ser referidas as ferramentas utilizadas durante todo o projeto, bem como as tecnologias empregues na aplicação de modo a facilitar as interações e operações realizadas pelos utilizadores.

3.4.1 Ferramentas utilizadas

No decorrer do projeto foram utilizadas diversas ferramentas com fins específicos e que se adequaram às diferentes fases do projeto, nomeadamente:

- *Android Studio*
- *Apache Axis2*
- *Apache Tomcat*
- *draw.io*
- *Eclipse*
- *GitLab*
- *Microsoft SQL Server*
- *SoapUI*

Nas primeiras fases do projeto utilizou-se a plataforma *draw.io*. O ***draw.io*** assenta numa plataforma do *Google Drive*, que permite a criação de diagramas, fluxogramas e esquemas a partir do *browser*. Esta ferramenta foi utilizada na fase de análise e desenho (*design*), para a criação de diagramas de fluxo, diagramas de casos de uso e diagramas de sequência. A escolha desta ferramenta deveu-se à facilidade de utilização e configuração das funcionalidades disponíveis e nos tipos de formatos disponibilizados para guardar os diagramas (Yuan *et al*., 2017).

Uma vez que foi desenvolvida uma aplicação em *Android*, foi essencial a configuração de um ambiente, que ajude nesse desenvolvimento. Para tal, a *Google* criou um *software*, que juntamente com uma linguagem de programação (*Java*), possibilita a criação de aplicações para *Android* (Yuan *et al*., 2017). O ***Android Studio*** corresponde a um *software* que cria um ambiente de desenvolvimento integrado (IDE), baseado no *software IntelliJ IDEA* e foi desenvolvido especificamente para *Android*. A ferramenta foi utilizada durante toda a fase de desenvolvimento, sendo das ferramentas essenciais durante todo o projeto (Zechner *et al*., 2016).

Visto que existiu a necessidade de criar um servidor local para serem implementados os serviços, foram utilizadas mais duas tecnologias: o *Eclipse* (para desenvolver os serviços) e o *Apache Tomcat* (para implementar os serviços, usando o *Apache Axis2*).

O ***Eclipse*** é um IDE para o desenvolvimento em *Java*, contudo suporta outras linguagens e *plugins*, tornando-o muito procurado na área do desenvolvimento de *software*. No projeto utilizou-se para a criação dos serviços *web*, devido ao maior conhecimento sobre esta ferramenta, já utilizada durante a parte teórica do Mestrado, em detrimento do *NetBeans* (Eclipse Foundation, 2018).

O ***Apache Tomcat*** corresponde a um servidor aplicacional *Java* mais usado nos últimos anos, chegando a representar 63,8% dos utilizadores em 2017. A sua popularidade foi conseguida graças à sua estabilidade e tornou-se um projeto sustentado pela *Apache Software Foundation* em paralelo com outros projetos de código aberto (*open source*) (Salnikov, 2017).

O ***Apache Axis2*** é um mecanismo para serviços *web*, atribuindo uma interface para os serviços disponíveis. Suporta várias versões do serviço SOAP (*Simple Object Access Protocol*) e possui um suporte integrado para o REST (*Representational State Transfer*). O *Apache Axis2* não é um substituto do *Tomcat*, dado que não é um servidor *web*, apenas permite que qualquer aplicação desenvolvida pelo *Axis2* possa ser implementada no *Tomcat* ou em qualquer servidor, desde que seja compatível (Apache Software Foundation, 2018).

Uma vez que o projeto foi desenvolvido por fases, onde cada fase correspondeu à realização de uma funcionalidade ou atualizações de funcionalidades, foi necessário o auxílio de um sistema que fizesse o controlo de versões e que unisse todas as funcionalidades desenvolvidas e atualizadas separadamente. Neste projeto foi utilizado o *GitLab*. O ***GitLab*** trata-se de uma plataforma que permite guardar os repositórios *online*, assente no *software Git*, que é um *software* de controlo de versões. Foi um sistema importante em toda a parte de desenvolvimento, devido à capacidade de controlar versões, permitir o desenvolvimento em paralelo e ter a capacidade de o repositório de ficheiros poder ser acedido e alterado a partir de qualquer lugar e de qualquer computador (Fedoseev *et al*., 2016).

Com a evolução exponencial da tecnologia, as empresas tiveram a necessidade de armazenar os seus dados em sistemas de gestão de bases de dados, devido ao acesso mais rápido aos dados e ao conhecimento que essas ferramentas permitem aos seus clientes, na manutenção de contratos e na construção de relações cliente/empresa. Neste projeto, o *software* para armazenar e gerir os dados do sistema foi o *Microsoft SQL Server*. O ***Microsoft SQL Server*** corresponde a um sistema de base de dados relacional (SGBD) desenvolvido pela *Microsoft*. Neste projeto, a principal função foi o armazenamento e recuperação de dados solicitados por aplicações de *software*, usando a linguagem SQL, que é suportado nas versões mais recentes do *Microsoft SQL Server* (Foster e Godbole, 2016).

Após as fases de análise, desenho e desenvolvimento, foi essencial efetuar testes, de modo a garantir a qualidade das funcionalidades prestadas. Com o intuito de testar os serviços *web*, foi utilizada a ferramenta *SoapUI*. O ***SoapUI*** é uma ferramenta para testes SOAP e REST. Possui uma interface gráfica de fácil utilização e com recursos de classe empresarial, permitindo que sejam executados testes funcionais de regressão ou de carga automatizada de modo fácil e rápido (SoapUI, 2018).

3.4.2 Tecnologias utilizadas

Um dos objetivos do projeto foi a utilização de novas tecnologias que permitissem modernizar e aumentar a usabilidade da aplicação final. Para este efeito, foram utilizadas três tecnologias:

- processo de autenticação com a utilização da **impressão digital**;
- possibilidade de os utilizadores efetuarem transferências bancárias através de um **código de barras** (**PDF417**);
- utilização de um *modem* GSM (*Global System for Mobile Communications*) para o **envio de SMS** (*Short Message Service*).

3.4.2.1 Impressão digital

O **sensor de impressão digital**, inicialmente associado a dispositivos *premium*, começou a tornar-se mais vulgar e a ser introduzido em dispositivos *Android* lançados no mercado. Visto que se trata de um acessório recente no mundo *mobile*, as versões mais antigas do sistema operativo *Android* não suportam esta funcionalidade de autenticação. Versões iguais ou superiores ao *Android* 6.0 (*Marshmallow*) permitem a utilização da impressão digital como método de autenticação em aplicações, existindo verificações a nível de *hardware* e *software* antes de efetuar a deteção da impressão digital (Hildenbrand, 2017).

Apesar de existirem diversos tipos de sensores de impressões digitais no mercado, ambos têm a mesma finalidade, que corresponde à representação matemática das particularidades de uma impressão digital. Esta representação é armazenada e reconhecida numa área separada de

um dispositivo *hardware* chamada TEE (*Trusted Execution Environment*). O TEE pode usar processador e memória próprios, possuindo também uma proteção de entrada/saída, sendo apenas permitido o acesso aos dados armazenados no TEE, caso este aprove (Hildenbrand, 2017).

Quando existe um registo de uma impressão digital num dispositivo *Android*, o sensor recolhe os dados da digitalização, transferindo-os para o TEE. Este cria um conjunto de dados de validação e um modelo da impressão digital criptografado, usando uma chave que apenas o TEE conhece. Por fim, os dados serão armazenados num local criptografado no interior do TEE ou noutra diretoria do dispositivo móvel, desde que seja criptografada (Hildenbrand, 2017).

Qualquer aplicação, que faça utilização da impressão digital como processo de autenticação, tem de superar o facto de nunca ter acesso aos dados armazenados no TEE nem aos dados reconhecidos pelo sensor. Todo o processo de autenticação por meio de impressão digital é efetuado pelo sistema operativo, que disponibiliza uma interface que deve ser implementada pelas aplicações. Estes mecanismos que guardam a impressão digital são assim por motivos de segurança: não queremos que as impressões digitais sejam distribuídas ou acedidas por terceiros, por serem informações altamente sensíveis. Assim, a informação é armazenada em *hardware* próprio para que seja ainda mais difícil a um agente malicioso aceder à informação, mesmo que consiga quebrar as barreiras de segurança do *software* (Hildenbrand, 2017).

3.4.2.2 Código de barras (PDF417)

O **PDF417** é um código de barras 2D baseado em códigos de barras empilhados. Os caracteres são codificados em palavras de código e podem ser numéricos ou alfanuméricos. Uma palavra de código compreende 17 módulos, que consistem, respetivamente, em 4 traços e 4 espaços. Foi criado em 1991 por Y. P. Wang na *Symbol Tecnologies* e os seus requisitos encontram-se no padrão ISO 15438. De entre as aplicações mais comuns destacam-se: aplicações logísticas; sistemas de transporte; documentação e identificação (TEC-IT, 2018).

Antigamente, os utilizadores despendiam algum tempo a preencher os requisitos mínimos para a realização de uma transferência, levando a que esta atividade se tornasse cansativa e morosa. Neste projeto, a utilização de um código de barras para a realização de uma transferência bancária foi adicionada aos tipos de transferências possíveis com o intuito de facilitar a realização desta funcionalidade. Neste contexto, existiu a necessidade de dinamizar a realização do processo de transferências bancárias, passando pela criação de um código de barras PDF417, para representar os dados de uma determinada transferência bancária (Figura 3.9). Das vantagens da sua utilização destacam-se: o armazenamento de informação a baixo custo de investimento e de manutenção. Contudo, apesar de teoricamente ser possível codificar

várias informações em códigos empilhados, tornaria o código complexo, dificultando a sua leitura.

Este tipo de código de barras utilizou-se ponderando uma perspetiva futura para a realização de transferências interbancárias, permitindo assim universalidade neste projeto.

Figura 3.9: Exemplo de um código de barras PDF417.

3.4.2.3 Envio de SMS

A aplicação por parte do servidor do cliente precisou também de um sistema que possibilitasse o **envio de SMS**. Após uma análise das diversas alternativas, totalizaram-se três: através de uma biblioteca do *Android* (`SmsManager`); através de uma API (*Application Programming Interface*), por exemplo *Twilio*; ou através do recurso a um *modem* GSM, tendo sido este último a escolha.

O `SmsManager` permite o envio de SMS utilizando os recursos do dispositivo onde está implementado o código. Para tal, basta ser introduzido o número de destino bem como o texto a ser enviado. Apesar de ser bastante fácil a sua utilização, não estava de acordo com o desenho da arquitetura, uma vez que o envio de SMS segundo a arquitetura implementada seria *server-side* (do lado do servidor do cliente). Em baixo segue-se um exemplo do envio de SMS utilizado a biblioteca do *Android* (`SmsManager`) (Developer Android, 2018).

```
SmsManager smsManager = SmsManager.getDefault();
smsManager.sendTextMessage("numero de telemovel", null,
        Texto a enviar", null, null);
```

A utilização de uma API para enviar SMS (por exemplo: *Twilio*) também possui limitações nesta área. A aplicação dependeria de uma API para efetuar uma operação, comprometendo o normal funcionamento da aplicação, caso a API cancelasse o seu funcionamento. Outras limitações descritas são os custos extra com o pagamento de uma taxa pela utilização dos serviços e a possibilidade de falhas no envio de SMS, devido a problemas de disponibilidade da API (Twilio, 2018).

A solução passou pela utilização de um *modem* GSM para o envio de SMS. Apesar do tempo despendido no desenvolvimento e compreensão sobre o seu método de funcionamento, levou a um maior controlo e garantia de que todas as SMS seriam enviadas. O *modem* GSM é uma tecnologia específica de *modem*. Trata-se de um dispositivo (onde se insere um cartão SIM) sem fios e com saída USB (*Universal Serial Bus*) para conexão com outro dispositivo (por

exemplo: computadores e *tablets* com adaptador). Recebe e descodifica o sinal digital que as operadoras transmitem para os aparelhos portáteis. Para que o servidor comunique com este *modem* é necessário que sejam enviados comandos AT (comandos *Hayes* – desenvolvidos em 1981 para comunicar com *modems*) com o objetivo de efetuar outras funcionalidades permitidas, que no caso do projeto foi o envio de SMS (Pandya e Shukla, 2012).

3.5 Rede de computadores

Em 1994 acreditava-se que os IPs iriam acabar, uma vez que o número de máquinas aumentou drasticamente e todas elas teriam de ser identificadas na rede para poderem comunicar entre si. Como teria de lhes ser atribuído um IP, levou a pensar-se que começariam a existir mais máquinas do que IPs e, com isto, muitas delas seriam incapazes de ter acesso à rede mundial. Com a finalidade de suprimir o problema da escassez de IPs, houve a criação de diversos protocolos (por exemplo: IPv6, NAT) e tipos de redes (por exemplo: privadas).

Foi criada a rede privada, que consiste numa rede que usa o espaço privado de endereços IP, seguindo padrões estabelecidos pela RFC 1918 para redes IPv4 e RFC 4193 para IPv6. Estes endereços dão a possibilidade de vários equipamentos comunicarem entre si dentro da rede privada, não fazendo parte da Internet. As redes privadas começam a ser comuns, uma vez que não há a necessidade de todos os dispositivos *hardware* possuírem um IP universalmente endereçável. Outra razão, apresentada inicialmente, deve-se ao número limitado de IPs públicos. Para solucionar este problema foi criado o IPv6. O IPv6 corresponde à versão mais atual do protocolo de Internet. Designada como a "nova geração do IP", tratou-se de um esforço do IETF (*Internet Engineering Task Force*) para solucionar problemas na Internet, tanto a escassez do IPv4 como a falta de segurança que o mesmo oferece (IPsec). Salienta-se também que o IPv6 é um protocolo que é implementado gradualmente e deve funcionar em paralelo com o IPv4, até o substituir definitivamente. Uma das principais limitações ocorre ao nível dos equipamentos antigos, pois estes não suportam o novo protocolo de Internet (Comer, 2016).

Os *switch* não conseguem encaminhar pacotes com IPs privados. Deste modo é necessário que exista um *router* para efetuar uma ponte de ligação entre a rede interna (privada) e a rede externa (pública). Usualmente os *routers* têm um serviço NAT (*Network Address Translation*) que permite a comunicação entre redes. O NAT consiste num protocolo que faz a tradução dos endereços IP e portas TCP da rede local para a Internet. Sempre que um pacote é enviado para a Internet, o *router* altera o IP/Porta da máquina para o seu IP/Porta, guardando o identificador do pedido mais dispositivo numa tabela de *hash*. Quando recebe a resposta, faz a operação inversa, modificando o IP/Porta para a do *hardware* específico, após verificação da tabela de *hash*.

O que faz com que não se consiga aceder de qualquer rede é o facto de o IP do servidor ser privado e de os *switch* não conseguirem encaminhar os pacotes para um IP privado (encaminhamento na rede).

Para possibilitar o envio de pacotes de qualquer rede para um servidor é possível efetuar de duas maneiras: através da aquisição (pós-pagamento) de um endereço IP público e associá-lo à máquina onde se encontram os serviços. Por outro lado, podemos migrar os serviços para uma máquina que já possua um IP público. Efetuando uma destas opções torna-se possível aceder aos serviços (servidor) através da aplicação conectada a qualquer rede.

3.6 Aplicações móveis

Atualmente, a população utiliza aplicações móveis para realizar diversas operações no seu quotidiano. Com esta evolução, as empresas apostam na criação, no desenvolvimento e na inovação *mobile*, com o objetivo de aumentar a proximidade com os clientes. De acordo com a abordagem de desenvolvimento, existem três tipos de aplicações (Silva *et al*., 2015):

- Aplicações *web*.
- Aplicações híbridas.
- Aplicações nativas.

A escolha do tipo de aplicação depende de inúmeros fatores, tais como, por exemplo (Gok e Khanna, 2013):

- Objetivo e público-alvo da aplicação;
- Sistema operativo;
- Funcionalidades;
- Período de tempo para o desenvolvimento da aplicação;
- Orçamento financeiro.

3.6.1 Aplicações *web*

O acesso a uma aplicação *web* é feito através de um *browser*. A *Web App* refere-se a aplicações baseadas num navegador criadas para dispositivos móveis, que podem ser conectados sem fios. São desenvolvidas maioritariamente em *JavaScript*, CSS e HTML5, isto é, em linguagens *web* (Gok e Khanna, 2013; Silva *et al*., 2015).

A sua escolha é feita em projetos de tempo e custos reduzidos. A principal vantagem é a capacidade de atualizar as aplicações sem instalação de *software* nos dispositivos móveis, e a compatibilidade dos navegadores entre plataformas torna-a prática, popular e dinâmica (Gok e Khanna, 2013).

Contudo, não permitem a utilização de outras funcionalidades do dispositivo móvel, como, por exemplo, a câmara fotográfica. Outra desvantagem é o facto de necessitarem de conexão à Internet e serem mais difíceis de encontrar, uma vez que não existe um sistema sistemático de pesquisa global, como acontece na *App Store* e *Google Play Store*. De facto, existem algumas aplicações nas lojas *mobile* que instalam no dispositivo um ícone com a única funcionalidade de

abrir o *browser* no URL (*Uniform Resource Locator*) da aplicação, mas que não é usado por todas as aplicações *web* (Gok e Khanna, 2013).

3.6.2 Aplicações híbridas

As aplicações híbridas apresentam um comportamento nativo associado a um desenvolvimento com tecnologias *web*. O seu desenvolvimento para os diversos tipos de dispositivos móveis requer diferentes *frameworks* (por exemplo, *React Native*) e tecnologias *web* (por exemplo, HTML5 e *JavaScript*) (Silva *et al*., 2015).

Este tipo de aplicação tem sido destacado ultimamente, uma vez que a parte principal da aplicação é desenvolvida através do recurso a tecnologias *web*, diminuindo o seu tempo de desenvolvimento e custo. No entanto, o facto de a parte da aplicação desenvolvida com tecnologias *web* correr sobre uma *framework* dedicada e não sobre o *web browser* nativo permite à aplicação aceder a componentes do dispositivo móvel impossíveis de aceder com aplicações *web* (por exemplo, a câmara fotográfica mencionada anteriormente). O padrão de *design* desta aplicação é aplicável a dispositivos móveis e ambientes *desktop* (Gok e Khanna, 2013).

3.6.3 Aplicações nativas

As aplicações nativas são desenvolvidas para um sistema operativo específico (*Android*, iOS e *Windows*), sendo descarregadas a partir de uma loja de aplicações, para os dispositivos móveis. Assim sendo, o desenvolvimento é feito em linguagens de programação específicas: *Java* para *Android* e *Objective-C* para iOS (Gok e Khanna, 2013; Silva *et al*., 2015).

Uma das vantagens da sua utilização é a permissão de utilização de todas as funcionalidades do dispositivo, por exemplo, câmara fotográfica, GPS, calendário e galeria de fotos (Gok e Khanna, 2013).

Contudo, a principal desvantagem é que se for necessário disponibilizar uma aplicação para os dois sistemas operativos mais utilizados (*Android* e iOS), é preciso desenvolver duas aplicações diferentes, e assim aumenta a duração e orçamento do projeto. Além disso, é necessário aguardar pela aprovação da aplicação nas lojas correspondentes e sempre que existam atualizações, terão de ser replicadas em todos os sistemas operativos (Gok e Khanna, 2013).

O uso de *frameworks* como o *Xamarin* permite a criação de uma aplicação apenas, que é posteriormente compilada para os vários sistemas operativos, obviando assim a necessidade de ter várias *code* bases para cada sistema operativo. Contudo as desvantagens são: o código não é tão corretamente otimizado para os sistemas operativos e a aplicação fica excessivamente grande, pois a *framework* incorpora código extra na aplicação mesmo que não seja necessário para o seu normal funcionamento (Alves, 2017).

Neste projeto utilizou-se este tipo de aplicação para o sistema operativo *Android*. Alguns dos motivos que potenciaram a sua escolha foram: rapidez e eficiência no desempenho e aplicações intuitivas e interativas, em termos de usabilidade.

3.7 A plataforma *Android*

Neste projeto foi desenvolvida uma aplicação móvel através da plataforma *Android*. De seguida, descreve-se e analisa-se este tipo de plataforma, pois torna-se uma ferramenta importante na programação de aplicações móveis.

O *Android* corresponde a uma plataforma aberta (*open source*) de *software* para o desenvolvimento de aplicações para dispositivos móveis (*smartphones*, *tablets* e *smartwatch*), suportada pela empresa de tecnologia *Google* (Pereira e Silva, 2009; Developer Android, 2018).

Criada pela *Android Inc* (constituída por diversas entidades do ramo informático), sendo esta adquirida pela *Google* em 2005, com o objetivo de entrar no mercado *mobile* e explorar todo o seu potencial, uma vez que os equipamentos móveis são cada vez mais valorizados no quotidiano da população (Developer Android, 2018).

Existem estudos que revelam que o *Android* se tornou o sistema operativo mais utilizado no mundo. Portugal segue a tendência, onde o *Android* já representa 31% do tráfego dos *sites*, observável na Figura 3.10 (Marktest, 2018).

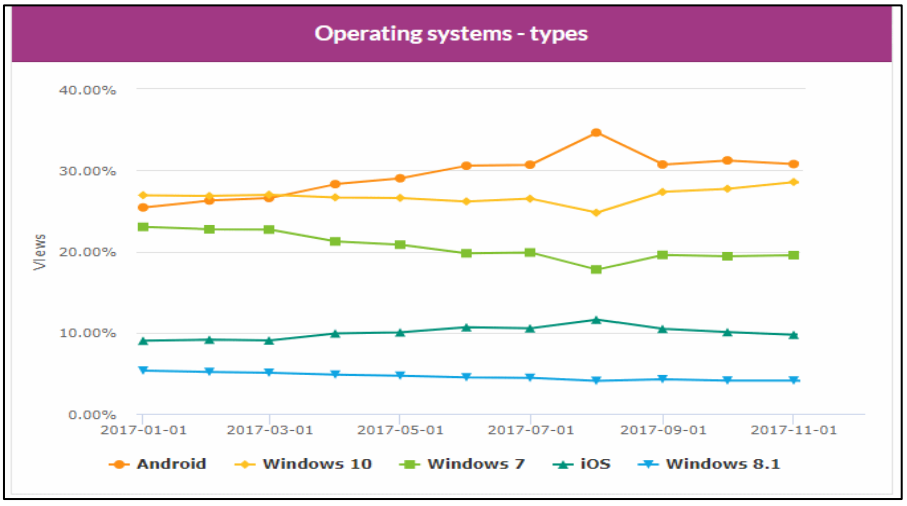


Figura 3.10: Utilização de sistemas operativos (janeiro – novembro 2017).
Marktest, 2018.

3.7.1 Arquitetura

O *Android* é constituído por um conjunto de aplicações nativas, uma camada que disponibiliza serviços (*middleware*) e um sistema operativo baseado no *Linux Kernel*. A Figura 3.11 representa a arquitetura da plataforma *Android*.

A camada inicial (*Applications*) engloba o conjunto de aplicações nativas em linguagem *Java*, de entre as quais: contactos, *browser*, cliente de email, galeria de fotos, programa para receção e envio de chamadas, entre outras. A segunda camada possibilita o acesso à *framework* de aplicações (*Application Framework*). Trata-se de um conjunto de funções específicas de *hardware*, onde o *Android* está a correr (por exemplo, *telephony manager* – acesso aos dados do telefone; GPS; câmara; áudio, etc.). A camada abaixo é composta por diversas bibliotecas (por exemplo, SQLite – Base de dados incluída), apresentadas ao programador através da camada superior (*Application Framework*). Possui também o *Android Runtime*, onde se encontram as bibliotecas *core* de linguagem *Java* e a máquina virtual *Java* do *Android* (*Dalvik Virtual Machine*). A diferença para a JVM (*Java Virtual Machine*) é ao nível do *bytecode*, pois em vez de transformar os arquivos em `.class`, como a JVM comum, transforma-os em `.dex` (formato *Dalvik bytecode*). Por último, encontramos o *Linux Kernel*, parte essencial na estrutura e funcionamento do *Android* e dispositivos (*hardware*). É responsável por algumas funcionalidades fulcrais, como a gestão da bateria, da memória, de processos e de segurança. É também responsável pela comunicação com o *hardware* (*Wi-Fi*, áudio, câmara, *display*) (Pereira e Silva, 2009; Developer Android, 2018).

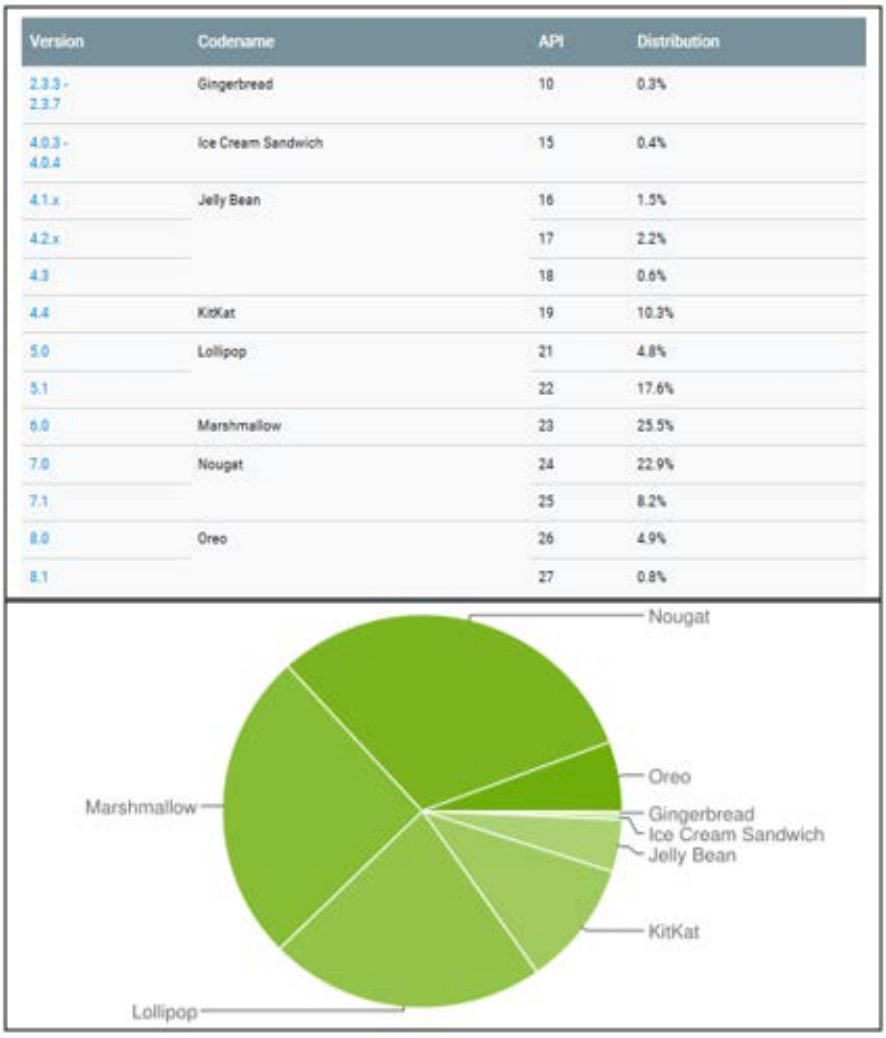
Figura 3.11: Arquitetura da plataforma *Android*. Developer Android, 2018.

O *Fuchsia* corresponde a um sistema operativo desenvolvido pela *Google*. Ao contrário dos sistemas operativos desenvolvidos anteriormente (*Chrome OS* e *Android*), que são baseados no *Linux Kernel*, este baseia-se no *microkernel* "*Zircon Kernel*", mais conhecido por *Mangenta*. Este produto corresponde a uma aposta da *Google* na "*Internet of Things*", podendo ser executado numa infinidade de dispositivos (Sinicki, 2018).

3.7.2 Versões do *Android*

A primeira versão do *Android* a ser comercializada foi a *Alpha* (*Android* 1.0) em setembro de 2008. Após esta data, existiram lançamentos internos em paralelo com a criação de *marketing* do produto. Entre 2008 e 2012 foram lançadas muitas outras versões (*Cupcake – Android* 1.5; *Eclair – Android* 2.0/2.1; *Gingerbread – Android* 2.3; *Honeycomb – Android* 3.0/3.1/3.2; *Ice Cream Sandwich – Android* 4.0 e *Jelly Bean – Android* 4.1/4.2/4.3) (Faustino, Calazans e Lima, 2017).

Segundo dados da *Google*, atualmente a versão mais utilizada é o *Nougat* (*Android* 7.0/7.1) com uma distribuição de 31,1%, sendo acompanhada pelo *Marshmallow* (*Android* 6.0) com 25,5% e *Lollipop* (*Android* 5.0/5.1) com 22,4%. A versão mais recente (*Oreo – Android* 8.0/8.1) possui uma taxa de utilização de 5,7% (Figura 3.12) (Developer Android, 2018).



Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.3%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.4%
4.1.x	Jelly Bean	16	1.5%
4.2.x		17	2.2%
4.3		18	0.6%
4.4	KitKat	19	10.3%
5.0	Lollipop	21	4.8%
5.1		22	17.6%
6.0	Marshmallow	23	25.5%
7.0	Nougat	24	22.9%
7.1		25	8.2%
8.0	Oreo	26	4.9%
8.1		27	0.8%

Figura 3.12: Dados relativos ao número de dispositivos de cada versão do *Android*. Em cima: distribuição relativa do uso das várias versões da API *Android*. Em baixo: distribuição relativa do uso das várias versões, agrupadas por nome de código. Developer Android, 8 de janeiro de 2018.

Uma das razões de existirem várias versões de *Android* ainda ativas no mercado, deve-se ao facto de os fabricantes de dispositivos bloquearem e não facultarem as versões mais recentes aos consumidores finais, tornando-se uma limitação de acesso a versões recentes e uma oposição aos objetivos da *Google*. A *Google* tal como qualquer empresa que lança um produto para o mercado, pretende introduzir qualidade, segurança, facilidade de utilização e inovação, sendo apenas possível com as atualizações e lançamentos de novas versões. Contudo, os dispositivos não atualizam automaticamente os sistemas operativos para versões mais recentes, fazendo com que um consumidor tenha a necessidade de adquirir um novo equipamento *hardware*, para dispor de todas as novidades das últimas versões.

3.7.3 Desenvolvimento de aplicações

A informação apresentada de seguida tem como base as referências, os conhecimentos científicos e os estudos descritos em Pereira e Silva, 2009; Faustino, Calazans e Lima, 2017; Developer Android, 2018, citadas nas referências bibliográficas.

3.7.3.1 Segurança

A segurança de um dispositivo *Android* é importante, uma vez que existem aplicações que acedem aos dados pessoais dos utilizadores e os utilizam nas próprias aplicações. Para prevenir que qualquer aplicação aceda de forma indevida aos dados pessoais, é introduzida uma permissão, sempre que a aplicação necessite de utilizar recursos do dispositivo. As permissões são definidas no ficheiro `AndroidManifest.xml` e, sempre que uma aplicação é instalada, irá pedir permissão ao utilizador (`<uses-permission>`). Por exemplo, se uma aplicação necessita de aceder à câmara para tirar uma fotografia, ou se precisa de aceder à Internet, esta terá de pedir permissão e só será instalada se o utilizador a aceitar. Em baixo, segue-se um exemplo de uma permissão de acesso à Internet.

```
<uses-permission android:name="android.permission.INTERNET" />
```

A partir do *Android* 6.0, as aplicações passaram a pedir permissões em *runtime*, porque muitos utilizadores instalam aplicações sem verificarem quais os dados pessoais a que irão aceder (contactos, fotografias, etc.).

3.7.3.2 *Activity*

A interface de uma aplicação é constituída por uma ou mais atividades (*Activity*). Estas são essenciais na comunicação entre a aplicação e o utilizador, pois detetam eventos criados e apresentam dados consoante o tipo de evento. Todas as aplicações têm de identificar as suas atividades no ficheiro `AndroidManifest.xml`, utilizando uma *tag* específica

(`<activity>`), bem como indicar qual a atividade que inicia a aplicação. Segue-se um exemplo da identificação de uma atividade no ficheiro `AndroidManifest.xml`.

```
<activity android:name=".MyActivity">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category
android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

Para permitir que uma atividade seja iniciada, deve-se implementar o método `onCreate(bundle)`, exemplificado em baixo. É neste método que é feita a utilização da função `setContentView(int)`, responsável por criar o ambiente gráfico com que o utilizador interage, e também onde tipicamente se podem criar referências aos elementos associados ao *layout* (`Button`, `TextView`, `EditText`, etc.), obtidos através do método `findViewById(int)`.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    Button button = (Button) findViewById(R.id.but);
}
```

O *Android* gere as atividades por meio de uma pilha, mostrando a atividade que está no topo (*running Activity*). Caso a atividade que se encontra no topo termine (*exit*), a atividade que se encontra imediatamente na posição posterior da pilha voltará para o topo e passará a ter tempo de execução, surgindo no ecrã do utilizador. A Figura 3.13 representa a visão geral do funcionamento de uma atividade.

De modo a facilitar a criação de *layouts* mais complexos e a sua gestão, uma atividade possibilita o uso de vistas (`Views` e `ViewGroup`) e fragmentos (`Fragment`). A classe `View` ocupa-se com uma determinada área retangular no ecrã e é responsável pelo desenho e manipulação de eventos. `View` é também a base dos *widgets*, que são usados para criar componentes de UI (*User Interface*) interativos (botões, campos de texto, etc.). A subclasse `ViewGroup` permite agrupar diversas `Views`, formando *layouts* mais complexos. Contudo, limitam o comportamento da aplicação ao nível da performance.

Um `fragment` controla uma parte da interface do utilizador ou o comportamento de uma aplicação, estando intimamente ligado à respetiva atividade. Apesar de ter um ciclo de vida próprio, este vai depender do ciclo de vida da sua atividade. A sua utilização é ampla, destacando-se o ajuste entre diversas dimensões de ecrãs.

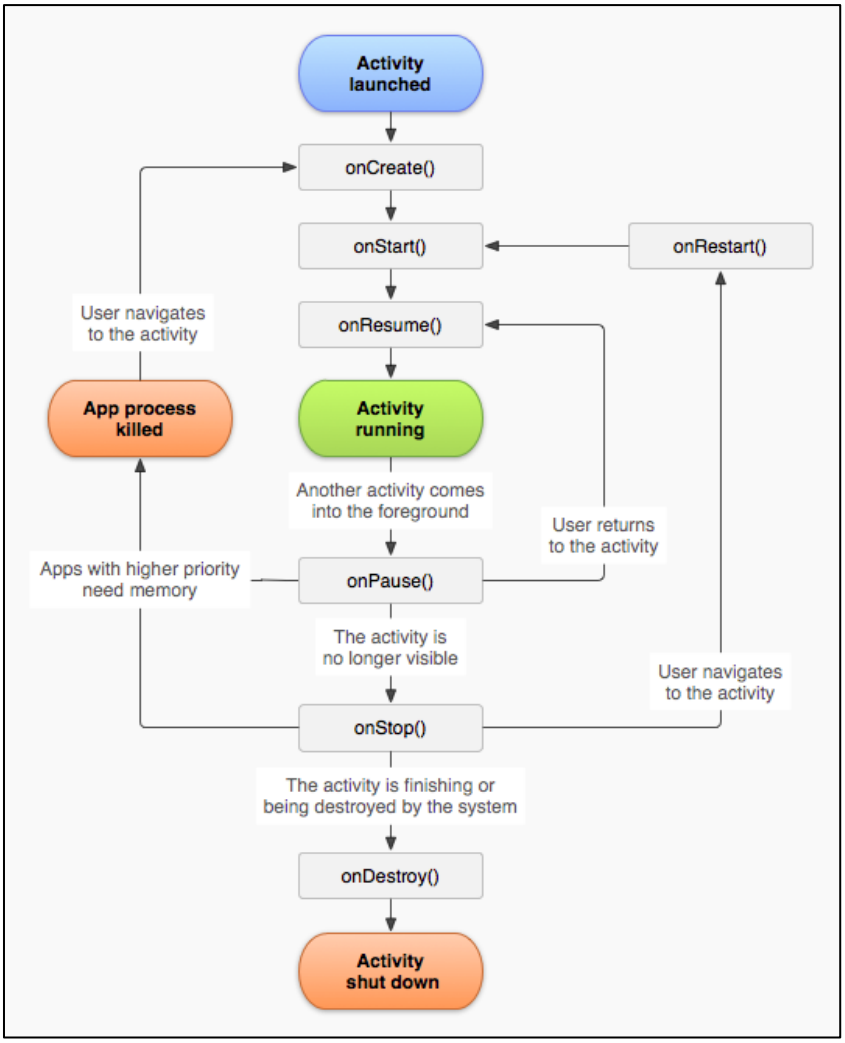


Figura 3.13: Ciclo de vida de uma atividade. Developer Android, 2018.

3.7.3.3 Comunicação entre *Activities* em *Android*

Como foi referido anteriormente, uma aplicação pode ser constituída por uma ou mais atividades. No entanto, a maioria das aplicações utilizam mais do que uma atividade para devolver ao utilizador todas as suas funcionalidades.

Desta forma, o *Android* teve a necessidade de criar um componente que permitisse passar de uma atividade para outra, enquanto a aplicação está em execução. Este componente é o `Intent`.

A classe `Intent` permite que uma atividade peça ao sistema operativo que inicie uma outra atividade através do método `startActivity(intent)`, possibilitando também o envio para qualquer `BroadcastReceiver` e a comunicação com um serviço em segundo plano, através do `startService(intent)`. Uma vez que as atividades comunicam dentro de uma aplicação, torna-se útil o envio de dados entre ambas. A classe `Intent` permite essa possibilidade, pois contém uma tabela que consiste numa coleção de chaves e valores a serem

enviados. A chave é do tipo `string`, sendo que o valor pode ser do tipo primitivo (`boolean`, `double`, etc.), `string`, `Bundle`, `Parceable` ou `Serializable`. Em baixo, segue-se um exemplo da comunicação entre duas atividades.

```
Intent intent = new Intent(ActivityUm.this, ActivityDois.class);
intent.putExtra("chave", true);
startActivity(intent);
```

Quando os dados são enviados e a atividade seguinte é iniciada, é possível ter acesso aos dados que lhe foram enviados, através do método `getIntent().getExtras()`, que poderá diferenciar consoante o tipo de valor que foi enviado. Em baixo, segue-se um exemplo para obter o valor enviado para a nova atividade.

```
boolean b = getIntent().getBooleanExtra("chave");
```

3.7.3.4 Estados de execução do *Android*

Quando um componente da aplicação é iniciado e a aplicação não tem outros componentes em estado de execução, o sistema *Android* cria um processo *Linux* para a aplicação com uma *single thread* de execução. Por norma, todos os componentes da mesma aplicação são processados no mesmo processo e *thread* (*Main Thread*), sendo que componentes de aplicações diferentes são executados em processos diferentes.

Apesar de as aplicações possuírem contextos de execução próprios, é possível associar componentes de uma determinada aplicação a processos diferentes através do ficheiro `AndroidManifest.xml`.

Atualmente, a população utiliza com frequência os dispositivos móveis e todas as aplicações que eles disponibilizam, fazendo com que existam muitas aplicações a serem executadas (a ocuparem memória) ao mesmo tempo. De modo a facilitar a utilização de diversas aplicações ao mesmo tempo, o *Android* mantém o máximo de tempo possível uma aplicação em memória (processo), sendo eliminada da memória, quando o dispositivo necessitar de mais espaço. Essa eliminação ocorre através de uma hierarquia, que o sistema possui, atribuindo níveis a cada processo. Existem quatro níveis de importância na hierarquia, sendo a sua importância apresentada por ordem decrescente:

1) *Foreground Process* – é um processo que corresponde ao que o utilizador está atualmente a executar. Este nível é o último a ser eliminado da memória, uma vez que está a ser utilizada para processar a aplicação, que está a interagir com o utilizador. Se o espaço de memória estiver limitado, este é removido, para não afetar o normal funcionamento do dispositivo móvel.

2) *Visible Process* – é um processo que não está a comunicar diretamente com o utilizador, apenas ilustra o *display*. Por exemplo, quando uma atividade chama um diálogo e permite que a atividade anterior fique visível para o utilizador, mas não focada (não recebe eventos do utilizador).

3) *Service Process* – é um processo que executa um serviço. Apesar desses serviços não estarem diretamente ligados ao que o utilizador está a realizar momentaneamente, estão relacionados com atividades, como ouvir música ou fazer um *download* de dados da rede.

4) *Cached Process* – é um processo que não tem nenhum componente da aplicação ativo. Apenas se mantém em memória por motivos de *caching*, facilitando o carregamento de uma aplicação quando é iniciada.

De forma a proporcionar uma boa experiência de utilização de uma aplicação aos consumidores, é essencial que a UI *Thread* nunca bloquei. Deste modo, é fulcral que sejam criadas novas *threads* para realizar operações não instantâneas. O sistema permite a criação de *Threads Background* ou *Worker Threads*, que permitem o processamento de ações não instantâneas. Contudo, estas não conseguem aceder ao *Android* UI *Toolkit*. Existe também a possibilidade de efetuar operações em *background* e publicar os resultados na *Main Thread*, através da classe `AsyncTask`.

3.7.3.5 *Software* de desenvolvimento e testes

O *Kit* de Desenvolvimento de *Software* (SDK) do *Android* inclui uma lista de ferramentas de desenvolvimento e compilação de aplicações *Android*. As aplicações desenvolvidas podem ser testadas rapidamente, porque podem serem criados diversos *Android Virtual Devices* (AVD) ou emuladores.

Atualmente o *Android Studio* é o ambiente aconselhado pela *Google* para o desenvolvimento *Android*, pois além de ser baseado no *IntelliJ IDEA* também apresenta novos recursos e melhorias em comparação com o *Eclipse* ADT (*Android Development Tools*).

Capítulo 4

Trabalho relacionado

4.1 Serviços bancários: enquadramento histórico

Os primeiros bancos surgiram no século XVIII A.C., na Babilónia e no Império Romano, através de empréstimos de padres a comerciantes. O banco mais antigo do mundo, ainda em funcionamento, é o Monte dei Paschi di Siena datado de 1472, em Itália (Choudhury, 2017).

Em Portugal, o desenvolvimento do sector bancário foi marcado pelas políticas liberalistas de Fontes Pereira de Melo, na segunda metade do século XIX. Os principais fatores foram a necessidade de financiamento por parte da indústria, agricultura e do Estado Português. O Banco de Lisboa, criado em 1821, foi o primeiro banco a existir no país, tendo sido substituído em 1846 pelo Banco de Portugal, que ainda hoje, regula as entidades bancárias existentes. Neste período, existiam mais de cinquenta instituições de crédito, localizadas maioritariamente em Lisboa e no Porto (Carvalho, 2013).

No Estado Novo, a quota de mercado dos depósitos e créditos nacionais concentrou-se nos grandes grupos financeiros, até então formados. Entre 1950 e 1974, a reorganização e regulamentação de algumas operações bancárias (depósitos à ordem, a prazo e com pré-aviso) permitiram a sofisticação no sector bancário. Este período ficou conhecido com os “30 anos de ouro da economia” (Mendes, 2002).

“Existem neste momento em Portugal mais de três dezenas de instituições bancárias, caixas de crédito e caixas económicas”, em funções ativas, distribuídas por todo o país (Rocha, 2017).

Em 2016, Carlos Costa, do Banco de Portugal, declarou que “o país tem cinco agências bancárias por dez mil habitantes, mas se calhar ainda é muito”. Segundo a análise da Figura 4.1 verifica-se que o sector diminuiu o número de agências em 24,9% e, consequentemente o número de empregados em 17,4%, no período de 2010-2016.

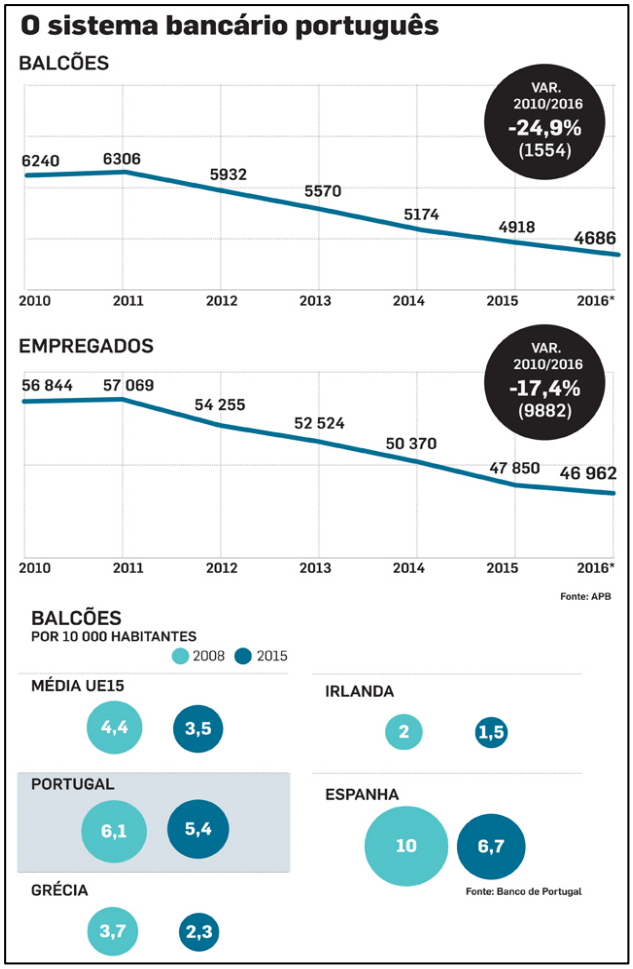


Figura 4.1: O sistema bancário português (2010-2016).
Banco de Portugal, novembro 2016.

4.2 Evolução tecnológica associada ao sector bancário

A evolução tecnológica e a crescente utilização de canais digitais são alguns dos problemas que este sector está a enfrentar. Numa análise do Banco de Portugal, liderado por Carlos Costa, defende que "as novas tecnologias trouxeram melhorias de eficiência e que essa dimensão pode salvar o futuro do sector".

4.2.1 Serviços bancários: ATM

A primeira ATM (*Automated Teller Machine*) foi instalada em 1967, em Londres, numa agência do *Barclays*, tendo sido inventada por Sheperd-Barron. Segundo a agência britânica *Reuters*, existem mais de três milhões de máquinas distribuídas pelo mundo (Reuters, 2017).

Em 1985, Portugal foi um dos últimos países da Europa a instalar este sistema, inicialmente com 12 terminais distribuídos por Lisboa e Porto. A empresa SIBS (Sociedade Interbancária de Serviços) gere a rede de ATM Multibanco, com cerca de 11 956 terminais no país, processando 75 milhões de operações bancárias mensalmente, no valor de 4,8 mil milhões de euros. Das inúmeras vantagens destacam-se:

- Disponibilidade de 24 horas por dia e 7 dias por semana;
- O *software* é compatível com todas as marcas e modelos de ATM, garantindo ao utilizador: fiabilidade, desempenho e segurança nas funcionalidades pretendidas (SIBS, 2018).

De acordo com a Figura 4.2, verifica-se que o sector bancário diminuiu o número de multibancos (ATM), no período de 2011-2017. Contudo, Portugal ocupa o primeiro lugar na zona Euro, no que respeita ao número de ATM por mil habitantes, com 1,38 ATM por mil habitantes face a uma média de 0,67 nos países da zona Euro.

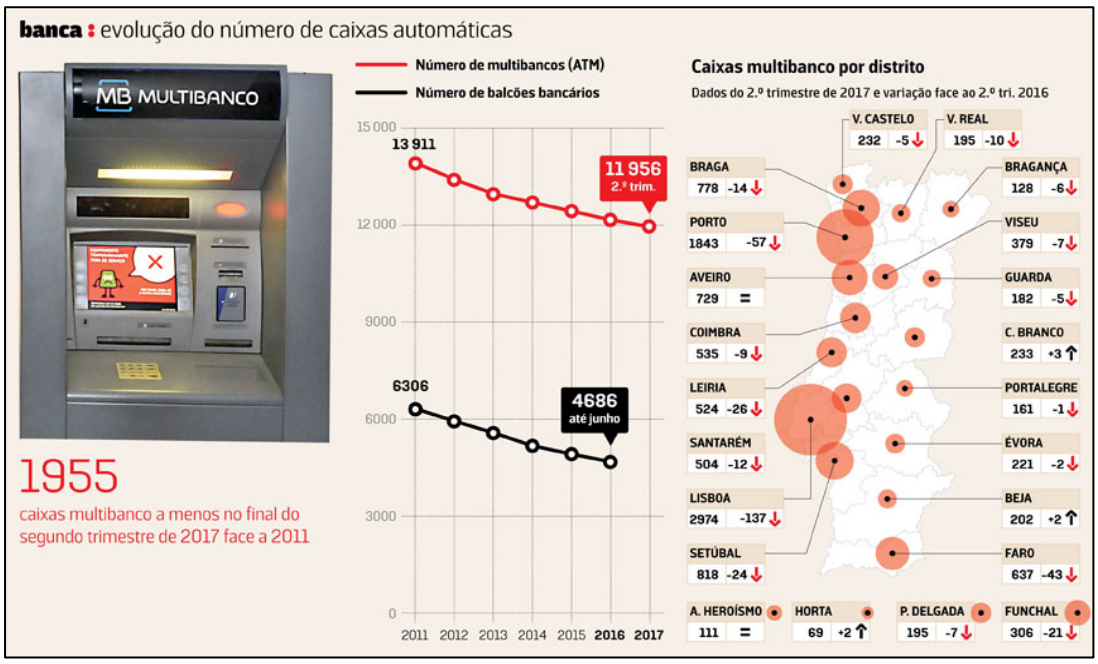


Figura 4.2: Evolução do número de caixas automáticas (2011-2017). Associação Portuguesa de Bancos, 2016; SIBS, 2018.

Atualmente a rede Multibanco tem um novo concorrente que, chegou a Portugal em maio de 2015. Trata-se de 300 caixas automáticas da ATM rede *Euronet Worldwide*, destinadas especialmente a estrangeiros. Têm como principais funções: levantamentos, transferências e sistema de pré-pagamentos (Rocha, 2017).

A evolução tecnológica proporcionou alterações significativas nos serviços bancários tradicionais, tendo surgido o *home banking*, *internet banking* e *mobile banking* (Liao *et al.*, 1999).

4.2.2 A evolução dos dispositivos móveis

Um dispositivo móvel (*handheld*) define-se como um sistema de pequenas dimensões, de modo a facilitar o seu transporte e manuseamento. É composto por dois componentes (*hardware* e *software*) que, em simultâneo, permitem que um utilizador realize tarefas de um modo rápido,

fácil, seguro e em qualquer local. O *input* e *output* combinam-se num ecrã táctil (*touchscreen*), em determinados dispositivos. Os dispositivos móveis mais comuns são: *smartphones*, PDA (*Personal Digital Assistants*), telemóveis e *tablets* (Weiss, 2002).

Entre 1947 e 1973, várias empresas do sector das telecomunicações desenvolveram testes tecnológicos e radiotelefónicos para a utilização pessoal de dispositivos móveis. Contudo, o primeiro modelo comercializado (Motorola DynaTAC 8000x) surgiu em 1983, nos EUA. A primeira geração de telemóveis (1G) teve um grande impacto na sociedade, pois foi o sinal de telefone analógico que permitiu essencialmente a comunicação por voz. No entanto, não possibilitava os serviços de *roaming* (Weiss, 2002; Shukla *et al*., 2013).

Na década de 90 do século XX, face às necessidades de melhoramento da qualidade nas chamadas de voz, surgiu a segunda geração (2G). Permitiu a substituição do sinal analógico para digital, com a tecnologia móvel GSM, onde passou a ser possível enviar e receber SMS – 1993, na Finlândia. Estes dispositivos reduziram as dimensões e começaram a consumir menos bateria. Em Portugal, o serviço móvel terrestre surgiu em 1989 e as primeiras operadoras móveis (TMN, Telecomunicações Móveis Nacionais SA e Telecom Portugal SA) entre 1991 e 1992 (Weiss, 2002; Shukla *et al*., 2013).

Os serviços da terceira geração (3G) iniciaram-se em 2001, no Japão, e chegaram à Europa em 2004 através da Vodafone. Com o aumento da transmissão de dados até 2 Mbps, as telecomunicações sem fios passaram a ser uma realidade, proporcionando a primeira grande experiência com a banda larga móvel. Os serviços de multimédia (vídeo, áudio, dados digitais, vídeo conferência), bem como outros serviços e aplicações baseadas na Internet, foram possíveis devido à tecnologia UMTS (*Universal Mobile Telecommunications System*). Desta forma, as funções estavam disponíveis em qualquer lugar, dependendo da cobertura da operadora móvel (Weiss, 2002; Shukla *et al*., 2013).

A geração 4G surgiu em 2010 e veio proporcionar maior velocidade, maior largura da banda, melhor cobertura e maior qualidade da rede. Distingue-se também pela rapidez de transmissão de dados e pela eficiência no acesso a serviços disponíveis na Internet, aliada à evolução dos dispositivos móveis, com ecrãs de dimensões superiores e de maior resolução. Esta geração funciona com a tecnologia LTE (*Long Term Evolution*) – tecnologia de transmissão de dados baseada na tecnologia WCDMA (*Wide-Band Code-Division Multiple Access*) e GSM. É também distinguida a WiMAX, que tem como objetivo promover a compatibilidade e interoperabilidade entre equipamentos baseados no padrão IEEE 802.16 (funciona em sistemas *Linux*) (Shukla *et al*., 2013).

A empresa *Huawei*, empresa multinacional de equipamentos para redes e telecomunicações sediada na China, pretende introduzir no segundo semestre de 2019, a tecnologia móvel 5G. De entre as inovações pretendidas, destacam-se o aumento da velocidade e da cobertura geográfica e diminuição da latência (Ferreira, 2018; Tannam, 2018).

Em Portugal, a rápida evolução e o aumento da utilização de dispositivos móveis (superior a 1 *per capita*), tanto para a atividade pessoal como profissional, levaram à inovação e desenvolvimento dos serviços financeiros móveis (Mallat *et al*., 2004; Laukkanen e Cruz, 2009).

4.2.3 Serviços bancários: *mobile banking*

O *mobile banking* remonta ao final da década de 1990, quando a *Paybox*, em colaboração com o *Deutsche Bank*, criou o primeiro serviço. Inicialmente, foi implementado e testado em países europeus: Alemanha, Espanha, Suécia, Áustria, Finlândia e Reino Unido. Entre os países em desenvolvimento, o Quénia foi o primeiro a introduzir este serviço, em 2007 (Shaikh e Karjaluoto, 2015). Segundo Laukkanen e Cruz (2009), a Finlândia é considerado o país europeu com a maior taxa de utilização de serviços móveis.

De acordo com os dados já apresentados na secção 4.1, Portugal apresenta condições propícias para o aumento da taxa de utilização do *mobile banking* (diminuição do número de agências e empregados bancários, número de dispositivos móveis superior a 1 *per capita*, comodidade e rapidez na realização de movimentos bancários). Salienta-se também que o *mobile banking* é a tecnologia mais recente introduzida no sector bancário em Portugal.

Pela análise da Figura 4.2 observa-se que entre os anos 2011-2017 tanto o número de multibancos (ATM) como o número de balcões bancários diminuiu. Em contrapartida, na Figura 1.2 observa-se que entre os anos 2014-2015, a utilização do serviço *mobile banking* através de aplicações aumentou para os 25,1% de utilizadores (Marktest, 2015; Associação Portuguesa de bancos, 2016; SIBS, 2018). Esta utilização ainda é reduzida comparativamente como outros países, pois em Portugal a taxa de população envelhecida é elevada e privilegia a relação entre cliente e gestor bancário.

Os consumidores finais procuram aplicações móveis, o que leva a que os serviços que existem procurem cada vez mais as empresas como a *Accenture* para desenvolverem estas aplicações. Desta forma, criam-se aplicações para prestar um serviço de excelência, fidelizar e reter clientes, podendo aceder aos serviços bancários em qualquer momento e lugar, de modo rápido, conferindo-lhe interatividade, ubiquidade e conveniência.

4.3 Aplicação já existente

Este projeto foi realizado com base numa aplicação já existente, em produção lançada no mercado. As diferenças entre a aplicação existente e a nova reformulação foram: interface, navegação, otimização de código e diminuição do número de interfaces, que influenciaram positivamente o tempo de resposta dos serviços.

Capítulo 5

Trabalho desenvolvido

5.1 Descrição geral do projeto

O objetivo do projeto desenvolvido ao longo deste estágio centrou-se no melhoramento e na facilitação da realização de transferências monetárias entre as contas dos clientes do banco. Em particular, um dos objetivos (como descrito na secção 1.2) foi a criação de uma aplicação móvel (a ser instalada pelos clientes do banco nos seus sistemas *Android*) que inclua a função de transferência monetária. Assim, e para atingir este objetivo, descrevem-se, de seguida, vários conceitos para facilitar a compreensão das secções seguintes.

Em primeiro lugar, um **cliente** define-se como um utilizador da aplicação móvel desenvolvida que seja cliente do banco que comissionou este projeto. Define-se também uma **transferência/transação** como um ato único que é originado por um cliente, e que tem como finalidade transferir uma quantidade monetária da conta desse cliente para a conta bancária de outro cliente. Estas transferências podem ser entre clientes do mesmo banco ou entre clientes de bancos diferentes. Ressalve-se aqui que o cliente de origem da transferência tem de ter uma conta no banco em questão. É também fundamental compreender a noção de **origem da transferência**. Por um lado, as transferências podem ser iniciadas pelo cliente que paga, sendo que este especifica a conta (que lhe pertence) de onde quer remover a quantia a pagar e a conta onde quer creditar essa quantia. Por outro lado, a aplicação suporta também a noção de uma transferência através de um código, sendo este criado pelo emissor da mesma. Esta funcionalidade foi a mais importante para o banco, uma vez que tornou o processo de transferência muito mais simples, não sendo necessário aos clientes digitar o número da conta de destino (ou o NIB ou IBAN respetivo), o que aumenta a facilidade de efetuar um pagamento, não só no que respeita ao tempo despendido pelo utilizador (que se vê assim dispensado do processo de digitar os vários caracteres que identificam uma conta), mas também no que respeita à capacidade do sistema detetar e corrigir erros (pois o utilizador, não tendo de escrever o código, não corre o risco de se enganar).

É também importante referir que, devido à necessidade de proteger a privacidade dos dados pessoais dos clientes, foi preciso ter atenção a várias noções de **segurança**. Para esse

efeito, foi elaborado um mecanismo de autenticação dos utilizadores que visa, por um lado, evitar que terceiros maliciosos se autentiquem como um utilizador do sistema, e, por outro, minimizar a carga e esforço mental dos utilizadores no processo de autenticação. Foi criado assim o conceito de um código pessoal, que é uma sequência de seis dígitos que o utilizador escolhe. Sempre que há necessidade de proceder à autenticação de um cliente, o sistema pede ao utilizador que digite um subconjunto dos dígitos da sequência inicial. O sistema permite que seja efetuada uma autenticação utilizando a impressão digital, facilitando assim o processo de autenticação, carga e esforço mental dos utilizadores.

Este capítulo apresenta todo o trabalho desenvolvido no projeto. Está dividido em 6 secções: **levantamento de requisitos** (secção 5.2), ***design* de *software* e base de dados** (secção 5.3), ***layouts*** (secção 5.4), **implementação** (secção 5.5), **testes** (secção 5.6) e **ambiente de produção** (secção 5.7).

5.2 Levantamento de requisitos

O projeto teve início a 2 de novembro de 2017, com o levantamento de requisitos, que correspondeu à primeira fase de desenvolvimento de *software*. É uma das fases essenciais na deteção de problemas que poderão ocorrer ao longo do projeto, reduzindo o tempo e custos de desenvolvimento. Permitiu também definir as funcionalidades essenciais que a aplicação suporta e a sua disponibilização no mercado *Android*.

O levantamento de requisitos foi realizado através da aplicação já existente (do cliente, não sendo desenvolvida pela *Accenture*), onde se retiraram: o controlo de fluxos, funcionalidades e características (requisitos funcionais e não funcionais), sendo posteriormente analisados, com o objetivo de encontrar problemas e falhas que pudessem vir a ser melhorados, de forma a facilitar a usabilidade da aplicação, garantido os níveis de segurança.

5.2.1 Dados obtidos

Apesar do cancelamento de algumas etapas do ciclo de desenvolvimento do projeto por parte do cliente, este disponibilizou a aplicação móvel e dados de testes, de modo a que fosse possível realizar as diversas funcionalidades com o objetivo de se efetuar um bom levantamento de requisitos. Este projeto focou-se em transações bancárias, portanto apenas serão apresentados diagramas referentes a esta funcionalidade. Os dados obtidos estão divididos em controlo de fluxo, funcionalidades e características de tarefas, após um reconhecimento do funcionamento da aplicação.

5.2.1.1 Controlo de fluxo

A realização destes diagramas permitiu analisar os aspetos dinâmicos da aplicação: a forma como as atividades (ver secção 3.7.3.3) se ligam durante a realização de uma

funcionalidade, os processos de autenticação do utilizador e a estrutura e apresentação da informação no ecrã.

A Figura 5.1 e Figura 5.2 apresentam os diagramas de fluxo referentes a transferências entre contas associadas à mesma instituição bancária (contas do mesmo banco) e entre contas associadas a diferentes instituições bancárias (contas interbancárias), respetivamente.

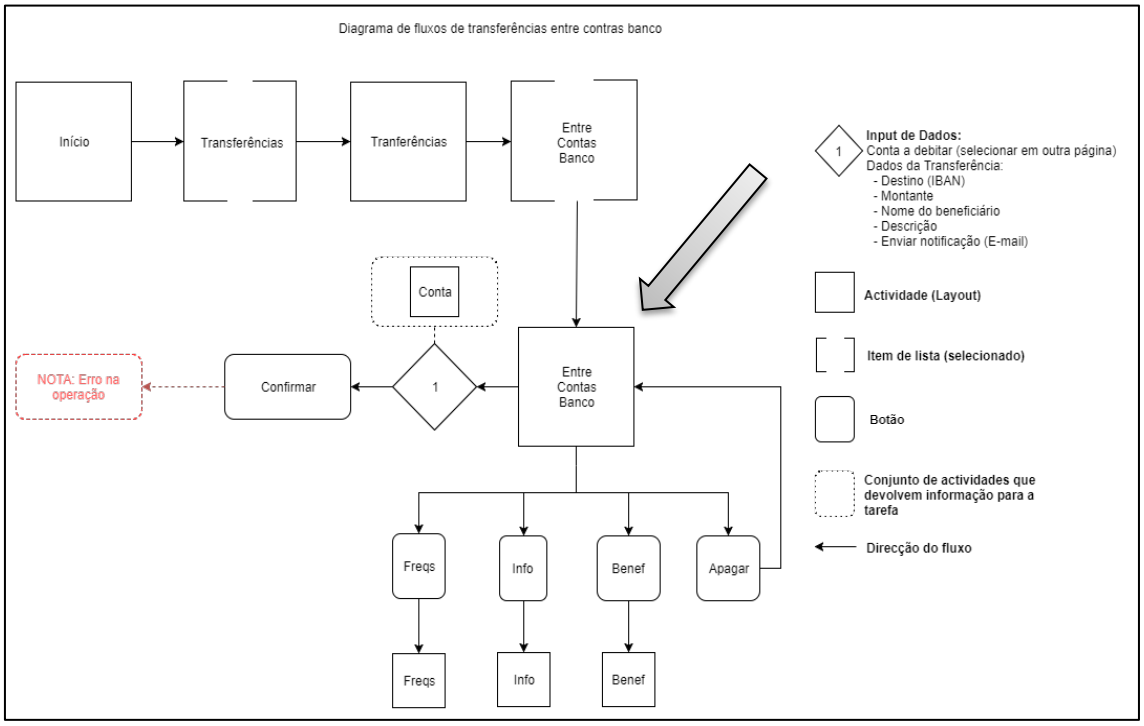


Figura 5.1: Diagrama de controlo de fluxo de transferências entre contas do mesmo banco.

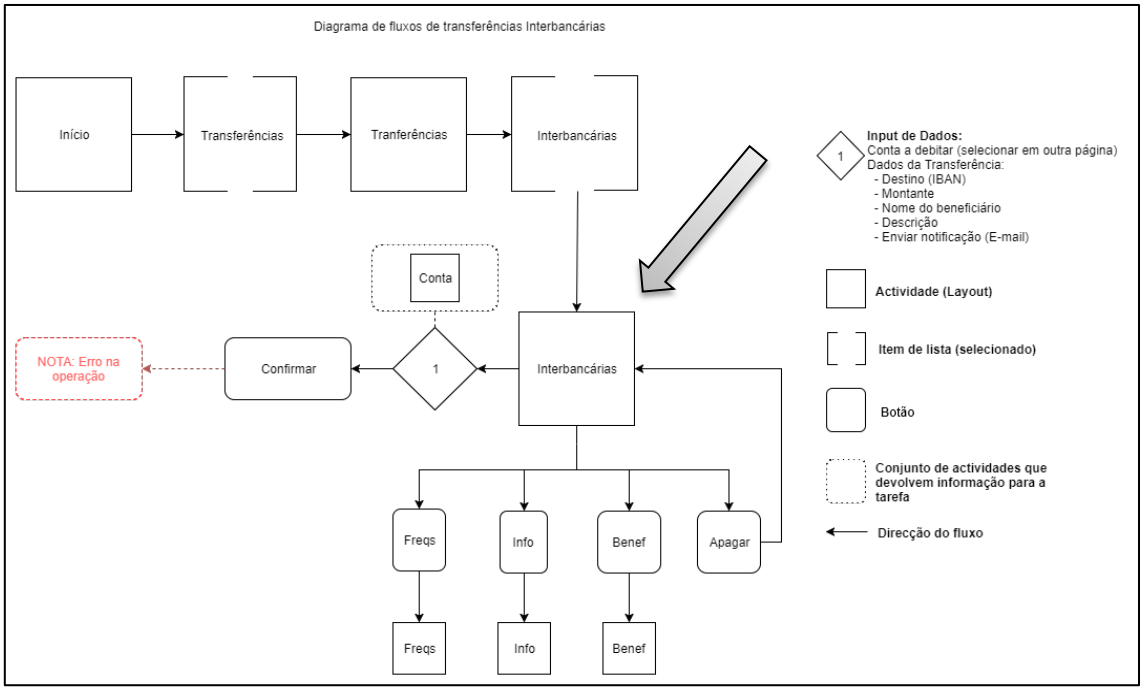


Figura 5.2: Diagrama de controlo de fluxo de transferências entre contas interbancárias.

Foram analisados estes dois casos, uma vez que existiam diferenças ao nível da funcionalidade (diferença marcada na figura através de uma seta). Cada um corresponde a uma funcionalidade diferente e, por isso, toda a navegabilidade e a parte lógica da aplicação são próprias e únicas. São úteis para avaliar o esforço despendido pelos utilizadores realizarem uma transação, bem como para detetar erros que acontecem constantemente durante a operação, limitando a utilização.

5.2.1.2 Funcionalidades

A aplicação incorpora as diversas funcionalidades que uma instituição bancária permite (consultas, pagamentos, transações, etc.). Contudo, uma vez que este projeto englobou um foco específico, todo o trabalho desenvolvido foi elaborado em torno das transações.

A aplicação disponibilizada pelo cliente permitia que os utilizadores apenas conseguissem efetuar transações de duas formas (entre contas do mesmo banco e interbancárias), não fazendo recurso a nenhuma tecnologia ou método para facilitar a usabilidade, levando o utilizador a preencher os campos sempre que necessitasse de realizar uma transação.

5.2.1.3 Características das tarefas

As aplicações bancárias são sistemas que realizam operações com dados privados do utilizador, existindo a necessidade da implementação de políticas de segurança, descritas no Capítulo 3.

Com o estudo da aplicação disponibilizada pelo cliente, uma das características desta era o processo de autenticação. Sempre que era necessária a atualização ou o acesso a dados privados, efetuava-se o processo de autenticação para que a funcionalidade fosse concluída. O processo de autenticação era realizado através de diversos códigos, fazendo com que o tempo da realização da tarefa aumentasse.

5.2.1.4 Limitações da aplicação existente

Após uma análise mais detalhada da aplicação já existente identificaram-se algumas limitações, nomeadamente:

- Dificuldade em aceder a determinados dados (início de sessão na aplicação);
- Esforço excessivo para a realização de tarefas;
- Processo de autenticação complexo e extenso;
- Elevados tempos de resposta do sistema;
- Excessivas falhas e erros no decorrer de uma tarefa;
- Fluxo de atividades complexo que requer demasiados ecrãs para realizar tarefas;

- Limitação na utilização de tecnologias.

Estes problemas e falhas provocavam alguma dificuldade na utilização da aplicação, e consecutivamente no quotidiano da população, uma vez que uma aplicação móvel visa facilitar a realização de tarefas na vida da população alvo.

5.2.2 Requisitos funcionais

A especificação de requisitos, tal como o levantamento, foi uma das componentes mais importantes da fase inicial, já que uma má especificação de requisitos poderia levar a falhas e atrasos no decorrer do projeto. Permitiram descrever o que o produto desenvolvido devia fazer ou qual o seu comportamento em determinadas alturas. Foram identificados catorze requisitos funcionais apresentados na Tabela 5.1.

Tabela 5.1: Requisitos funcionais.

RF01	Login do utilizador
O sistema deverá suportar um sistema de login (número de adesão e password). **Observações**: O utilizador é identificado através de um número de adesão e uma password pelo sistema.	
RF02	Realizar uma transferência entre contas do mesmo banco
O sistema deverá efetuar a realização de transferências entre contas do mesmo banco. **Observações**: Apenas realizará a transação após todos os campos estarem devidamente preenchidos.	
RF03	Realizar uma transferência interbancária
O sistema deverá efetuar a realização de transferências entre contas de bancos diferentes. **Observações**: Apenas realizará a transação após todos os campos estarem devidamente preenchidos.	
RF04	Criar uma transferência por código
O sistema deverá criar um código com os dados da transferência. **Observações**: Permite criar um código com os dados da transferência introduzidos pelo titular da conta de origem (conta de origem, montante, descrição e email).	
RF05	Efetuar uma transferência por código
O sistema deverá efetuar uma transferência por código ainda disponível. **Observações**: Realiza uma transferência através de um código e depois de o titular da conta de destino indicar a conta para onde deve ser realizada a transferência. A transação apenas deve ser realizada caso o código e o saldo da conta de origem esteja disponível.	
RF06	Visualizar transferências frequentes
O sistema deverá mostrar a lista de transferências que já foram realizadas anteriormente por ordem de frequência. **Observações**: Deverá mostrar uma lista com as transferências realizadas por ordem de frequência, sendo essa ordem calculada através do número de vezes que cada transferência foi realizada (usando a mesma conta de destino e o mesmo montante).	
RF07	Visualizar transferências beneficiário
O sistema deverá mostrar a lista de transferências que já foram realizadas anteriormente por ordem de beneficiário. **Observações**: Deverá mostrar uma lista com as transferências realizadas por ordem de beneficiário, sendo essa ordem calculada através do número de vezes que o beneficiário de cada transação foi destinatário.	

RF08	Enviar comprovativo de transferência por email
O sistema deverá permitir enviar um comprovativo de transferência para o email do titular da conta de origem. **Observações**: Deverá enviar um comprovativo com os dados da transferência realizada e da conta, após a realização da transferência. Só deverá ser enviado caso o utilizador tenha ativado essa opção.	
RF09	Enviar um código de transferência por email
O sistema deverá enviar para o email do destinatário o código de transferência. **Observações**: Deverá enviar o código criado para o email indicado pelo titular do código, caso o mesmo tenha ativado essa opção.	
RF10	Enviar um código numérico por SMS
O sistema deverá enviar um SMS com um código numérico. **Observações:** Deverá enviar um código numérico (nove dígitos) através de um SMS para facilitar na escolha do código pessoal de cada utilizador.	
RF11	Apagar um código de transferência
O sistema deverá permitir apagar um código de transferência (ainda ativo) que já não seja necessário. **Observações:** Deverá apagar um código de transferência que já não tenha interesse para o utilizador.	
RF12	Enviar um código numérico por email
O sistema deverá permitir o envio de códigos numéricos (nove dígitos) por email, durante o processo de autenticação duvidosa. **Observações:** Sempre que o sistema deteta uma autenticação duvidosa, deverá enviar um código numérico por email.	
RF13	Ler um código de transferência
O sistema deverá permitir a leitura de um código de transferência. **Observações:** Deverá efetuar a leitura do código de transferência através de um leitor digital ou através da imagem. Apresentando ao utilizador o titular que criou o código, a descrição (motivo para a criação daquele código) e montante.	
RF14	Autenticar um utilizador
O sistema deverá autenticar o utilizador através de um código pessoal ou por meios biométricos (impressão digital). **Observações:** Deverá autenticar o utilizador sempre que este aceda à sua conta (efetuar o login na aplicação) e realize transferências (entre contas do mesmo banco, interbancárias e código de transferência).	

5.2.3 Requisitos não funcionais

Como o sistema necessita de um ambiente de produção, devem ser considerados os requisitos incluídos neste ambiente. Os requisitos não funcionais não caracterizam diretamente o sistema, mas sim o ambiente onde será implementado, sendo um ponto essencial para resolver diversas questões, nomeadamente questões de disponibilidade. Os requisitos não funcionais são apresentados na Tabela 5.2, tendo sido elaborada com base nas características de um sistema que será utilizado por diversos utilizadores em simultâneo e que esteja em constante atualização.

Tabela 5.2: Requisitos não funcionais.

Requisitos não funcionais	
Confidencialidade	Dado ser essencial o recurso a dados do utilizador em diversos componentes da arquitetura e métodos de autenticação, deve haver garantias da confidencialidade dos mesmos.
Disponibilidade	O sistema deve ser o mais eficiente possível a responder, sendo que a média no tempo de resposta não deverá exceder os 3 segundos.
Extensibilidade	O sistema deve permitir a adição de novos elementos na arquitetura e funcionalidades no futuro, não requerendo muito esforço por parte da equipa técnica.
Modificabilidade	O sistema deve permitir que sejam efetuadas modificações em diversos elementos sem alterar o normal funcionamento do sistema.
Portabilidade	O sistema deve ser compatível com várias versões *Android*, uma vez que vai ser desenvolvido para esta plataforma.
Segurança	O sistema deve estabelecer uma comunicação segura entre os diversos componentes da rede.
Testabilidade	Deve ser possível testar os diversos componentes individualmente.
Usabilidade	A utilização por parte dos consumidores finais deve ser simples, sem necessidade de esforço de aprendizagem na realização de tarefas.

5.2.4 Casos de uso

Um caso de uso efetua uma especificação do comportamento de uma funcionalidade, demonstrando detalhes, restrições, premissas e diretivas pertinentes à funcionalidade (compare-se com os requisitos funcionais, que se referem a uma função a que um *software* deverá atender ou realizar, podendo ser associado um ou mais requisitos funcionais a uma funcionalidade).

Com o objetivo de facilitar a compreensão das principais funcionalidades por parte do cliente, foi necessária a criação do diagrama de casos de uso (Figura 5.3). O diagrama de casos de uso descreve de forma acessível e clara as principais funcionalidades que o sistema deve suportar. Tem como objetivo o auxílio da comunicação entre os diversos elementos do projeto e o cliente, descrevendo as funcionalidades do ponto de vista do utilizador.

Para facilitar a compreensão, tanto para o cliente como para as etapas seguintes, é necessário fazer uma descrição mais detalhada de cada caso de uso. Deste modo, a descrição dos casos de uso segue a seguinte estrutura: ator principal, interesses, pré-condições, pós-condições, cenário principal de sucesso e extensões. No Anexo A encontram-se os casos de uso deste projeto descritos detalhadamente na Tabela 5.3.

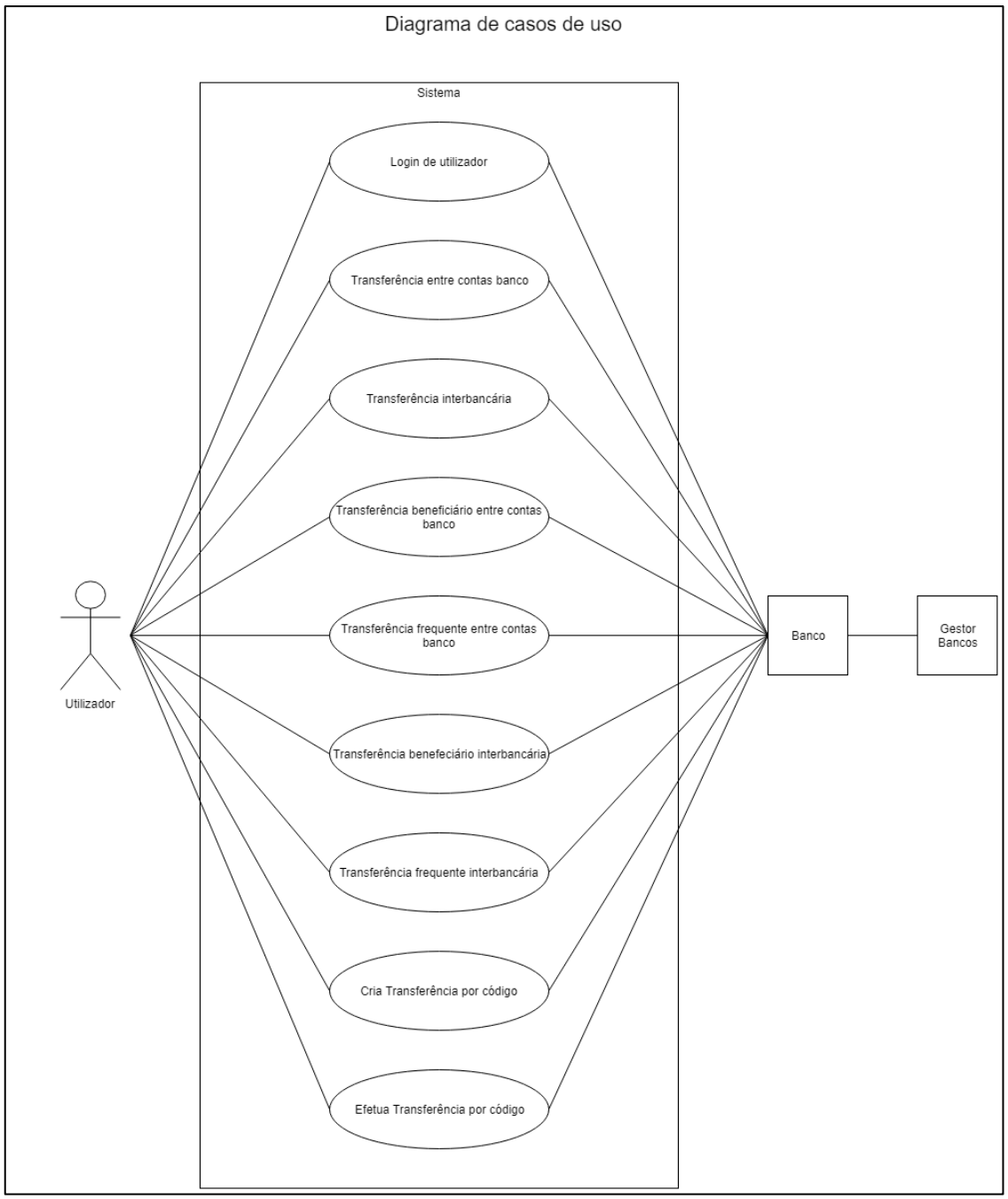


Figura 5.3: Diagrama de casos de uso.
(https://gitlab.com/Henry15/Imagens_da_tese/blob/master/Diagrama_de_casos_de_uso/Diagrama_caso_uso.png).

Tabela 5.3: Descrição dos casos de uso.

Caso de uso	Descrição	Anexo
UC1	Login de utilizador.	A.1
UC2	Transferência entre contas do mesmo banco.	A.2
UC3	Transferência interbancária.	A.3
UC4	Transferência beneficiário entre contas do mesmo banco.	A.4

Caso de uso	Descrição	Anexo
UC5	Transferência frequente entre contas do mesmo banco.	A.5
UC6	Transferência beneficiário interbancária.	A.6
UC7	Transferência frequente interbancária.	A.7
UC8	Cria Transferência por código.	A.8
UC9	Efetua Transferência por código.	A.9

5.3 *Design* de *software* e base de dados

Com o levantamento de requisitos já numa fase avançada, deu-se início à etapa de *design* de *software* e de base de dados. O produto resultante da fase anterior foi essencial para esta etapa, nomeadamente os casos de uso, onde foi descrito o comportamento das funcionalidades que foram implementadas posteriormente.

Esta secção encontra-se dividida em "***design* de *software***" (secção 5.3.1, onde é feita uma descrição detalhada do comportamento das ações, recorrendo a diagramas, contratos e modelos e, com isso, estabelecer uma ligação com a fase de implementação, "**desenho da base de dados**" (secção 5.3.2), onde é apresentada a lógica do modelo de dados, e "**arquitetura de *software***" (secção 5.3.3), onde se descreve a arquitetura em camadas, de sistema e de lógica.

5.3.1 *Design* de *software*

O *design* de *software* corresponde ao processo de passagem do espaço do problema para o espaço da solução. É no desenho que se encontra e elabora uma solução para o problema anteriormente encontrado, de modo a satisfazer os requisitos. Com a finalidade de conceber a melhor e mais elaborada solução, foram criados diagramas de sequência, contratos das operações, modelo de domínio, diagramas de interação e modelo de classe.

5.3.1.1 Diagramas de sequência

Os diagramas de sequência representam a sequência de mensagens passadas entre os diversos objetos do sistema, mostrando também ao longo do tempo, o conjunto de objetos que contribuem no processamento de uma determinada tarefa. Foram elaborados com o auxílio da descrição dos casos de uso (secção 5.2.4) e dão maior relevância aos objetos utilizados e à sequência temporal na troca de mensagens.

Neste projeto foi efetuado um diagrama de sequência para cada caso de uso apresentados no Anexo B e descritos na Tabela 5.4.

Tabela 5.4: Descrição dos diagramas de sequência.

Caso de uso	Descrição	Anexo
UC1	Login de utilizador.	B.1

Caso de uso	Descrição	Anexo
UC2	Transferência entre contas do mesmo banco.	B.2
UC3	Transferência interbancária.	B.3
UC4	Transferência beneficiário entre contas do mesmo banco.	B.4
UC5	Transferência frequente entre contas do mesmo banco.	B.5
UC6	Transferência beneficiário interbancária.	B.6
UC7	Transferência frequente interbancária.	B.7
UC8	Cria Transferência por código.	B.8
UC9	Efetua Transferência por código.	B.9

5.3.1.2 Modelo de domínio

Esta secção descreve a solução que visa resolver o problema encontrado durante a fase de análise. Para tal, foi criado um modelo de domínio. O modelo de domínio corresponde a um glossário do projeto, onde são identificados os termos utilizados e definidas as relações entre os mesmos. Faz uma representação visual de classes conceituais, ou objetos do mundo real, num determinado domínio do problema. Recorre à notação UML (*Unified Modeling Language*), permitindo ilustrar um conjunto de diagramas de classes, nos quais não se definem operações.

No Anexo C encontra-se o modelo de domínio representado por um diagrama em UML para os diversos tipos de transferências (entre contas do mesmo banco, interbancárias e por código), para o gestor de sessões, para as contas e para os clientes. Segundo esta figura destacam-se várias relações importantes no domínio, entre as quais: um cliente pode possuir mais do que uma conta; uma conta pode ser gerida por mais do que um cliente; um cliente pode ter uma ou mais transferências realizadas e um cliente pode conter nenhuma ou uma transferência em processamento. As restantes classes apresentadas no modelo de domínio (`BancoSoftware`, `Banco`, `CatalogoCliente`, `CatalogoConta`, etc.) encontram-se descritas na secção 5.3.1.5.

5.3.1.3 Contratos das operações

Com a finalização dos diagramas de sequência e do modelo de domínio, retiraram-se as operações que o sistema vai implementar, com o objetivo de evitar que as tarefas atribuídas às operações se tornem redundantes e inconsistentes. É fulcral que as operações estejam bem documentadas de modo a facilitar o ciclo de implementação.

Um contrato especifica o comportamento esperado para cada operação correspondente a um evento do sistema. Descrevem modificações detalhadas em objetos num modelo de domínio fazendo uso de uma pré e pós-condições, fornecendo assim mais detalhe sobre o efeito das operações no sistema implícito nos casos de uso. Como forma de documentação do projeto desenvolvido, os 29 contratos das operações possíveis estão descritos no Anexo D. Para cada

contrato são apresentados o nome da operação, os casos de uso associados à operação (*Cross References*), e as Pré-condições, Pós-condições e Parâmetros.

5.3.1.4 Diagramas de interação

Um diagrama de interação representa as interações existentes entre os elementos do sistema, mostrando os possíveis estados do sistema assim como as operações que são executadas para que o sistema modifique o seu estado. Apresenta os objetos das classes que participam na interação de uma operação. Mostram também as associações, as mensagens e respetiva ordem ocorridas entre os objetos. Foram elaborados através do modelo de domínio e de acordo com os contratos das operações e estão representados no Anexo E e descritos na Tabela 5.5. Informa-se ainda que os diagramas do E.1 – E.7 são referentes às operações do login do utilizador, enquanto que os diagramas do E.8 – E.27 se referem às operações de transferências.

Tabela 5.5: Descrição dos diagramas de interação.

Operação	Descrição	Anexo
login de utilizador	O sistema identifica o utilizador que está a efetuar o login.	E.1
número de telemóvel	O sistema cria um código SMS, armazena-o e prepara-o para o enviar para o número de telemóvel introduzido pelo utilizador.	E.2
código pessoal	O utilizador escolhe o seu código pessoal e confirma que recebeu o código SMS.	E.3
escolha de posições para autenticação	O sistema escolhe as quatro posições do código pessoal que o utilizador vai inserir.	E.4
autenticação usando código pessoal	O utilizador autentica-se utilizando as quatro posições do seu código pessoal.	E.5
autenticação usando sensor	O utilizador autentica-se usando a impressão digital.	E.6
autenticação usando código de email	O sistema cria um código, envia-o para o email do utilizador e efetua a sua leitura para proceder à autenticação do utilizador. Acontece sempre que o sistema deteta uma autenticação duvidosa.	E.7
transferência entre contas do mesmo banco	O utilizador escolhe o tipo de transferências entre contas do mesmo banco.	E.8
visualização de contas cliente	O sistema mostra as contas associadas ao utilizador autenticado.	E.9
confirmação de transferência	O sistema valida os dados introduzidos pelo utilizador e preenche os campos da transferência corrente.	E.10
escolha de autenticação com código	O sistema escolhe as quatro posições do código pessoal que o utilizador terá de inserir em processos de transferências.	E.11
autenticação com o código pessoal	Realização de uma transferência após autenticação, utilizando as quatro posições do código pessoal.	E.12
autenticação com sensor	Realização de uma transferência após autenticação, utilizando a impressão digital.	E.13

Operação	Descrição	Anexo
autenticação com código de email	Processo que cria um código, envia-o para o email do utilizador e efetua a sua leitura para proceder à realização da transferência. Acontece sempre que o sistema deteta uma autenticação duvidosa.	E.14
escolha de transferência interbancária	Escolhe o tipo de transferências interbancárias.	E.15
mostra transferências beneficiário entre contas do mesmo banco	Mostra as transferências já realizadas entre contas do mesmo banco ordenadas pelos beneficiários que recebem mais transações.	E.16
mostra transferências frequentes entre contas do mesmo banco	Mostra as transferências já realizadas entre contas do mesmo banco ordenadas pela frequência que ocorrem.	E.17
mostra transferências beneficiário interbancárias	Mostra as transferências interbancárias já realizadas e ordenadas pelos beneficiários que recebem mais transações.	E.18
mostra transferências frequentes interbancárias	Mostra as transferências interbancárias já realizadas e ordenadas pela frequência que ocorrem.	E.19
confirma transferência de uma já existente	O sistema associa a transferência corrente à selecionada pelo utilizador.	E.20
cria transferência por código	Escolhe o tipo de transferências por um código.	E.21
cria código de transferência	Preenche os dados da transferência com os dados que o cliente indicou. Prepara a transferência para ser associada a um código.	E.22
cria código de transferência com autenticação usando código pessoal	Cria o código de transferência e associa-o à transferência corrente após autenticação através das quatro posições do código pessoal.	E.23
cria código de transferência com autenticação usando sensor	Cria o código de transferência e associa-o à transferência corrente após autenticação utilizando a impressão digital.	E.24
cria código de transferência com autenticação usando código de email	Cria o código de transferência e associa-o à transferência corrente após o sistema ter identificado uma autenticação duvidosa.	E.25
leitura de código de transferência	Lê o código associado a uma determinada transferência. Permite que o utilizador tenha acesso à entidade que criou o código, ao montante e à descrição da transferência, aceitando ou rejeitando a transferência.	E.26
confirma a transferência de código	Realiza a transferência após leitura de um código e introdução da conta onde será depositado o montante da transferência.	E.27

5.3.1.5 Modelo de classes

Com o objetivo de simplificar o conjunto de classes que o sistema necessita bem como as relações entre elas, foi desenvolvido o modelo de classes. O modelo de classes representa a estrutura e as relações das classes que servem de modelo para os objetos, e corresponde às classes efetivas de *Java* que foram implementadas no sistema, e também aos conceitos do modelo de domínio apresentado na secção 5.3.1.2. Nesta fase vai ser apresentado o modelo de

classes desenvolvido e implementado posteriormente na fase de implementação e feita uma descrição sucinta de todas as classes. O modelo de classes da aplicação móvel encontra-se descrito na Tabela 5.6 e representado no Anexo F.1 – Modelo de classes da aplicação móvel.

Tabela 5.6: Descrição do modelo de classes da aplicação móvel.

Classe	Descrição
`Banco`	Classe responsável por gerir as classes, "*main*" do domínio.
`SoftwareBanco`	Classe responsável por comunicar com os serviços remotos. Toda a comunicação com o exterior do dispositivo móvel passa por esta classe.
`CatalogoCliente`	Classe responsável por gerir os clientes da aplicação.
`CatalogoTransferencia`	Classe responsável por gerir as transferências da aplicação.
`CatalogoConta`	Classe responsável por gerir as contas da aplicação.
`Autenticacao`	Classe associada às operações que envolvem dados referentes ao processo de autenticação do cliente.
`GestorSessao`	Classe que permite gerir a sessão no sistema.
`Cliente`	Classe que identifica o utilizador e reconhece todas as suas operações no sistema. A classe `cliente` é reconhecida pelo sistema de duas formas. A primeira como um utilizador que tenta entrar no sistema e por isso está em processo de login (já foi identificado, mas ainda falta ser autenticado - `processoCliente`). Caso não avance no processo de autenticação, o utilizador nunca sairá desta forma de reconhecimento. A segunda forma é a que o sistema reconhece como um utilizador autenticado. Neste caso, o utilizador passou no processo de autenticação e, desta forma, o sistema passou a reconhecer o utilizador como um cliente autenticado (`authCliente`).
`Conta`	Classe que representa as contas bancárias no sistema. Cada conta bancária equivale a um objeto da classe `Conta`.
`Seguranca`	Classe responsável por garantir a segurança com o exterior do dispositivo móvel. Sempre que é necessário existir a transferência de dados para o exterior, é utilizada a classe para encriptar os dados. No modelo de classes, a utilização desta classe foi simplificada para facilitar a compreensão do mesmo.
`Transferencia`	Classe que representa uma transferência no sistema. Cada transferência é representada por um único objeto no sistema.
`Codigo`	Classe que estende a classe `Transferencia`. É um dos tipos de transferências que o sistema permite. Responsável por transferências que se realizem através de um código.
`EntreContasBanco`	Classe que estende a classe `Transferencia`. Responsável por transferências que acontecem entre contas bancárias do mesmo banco.
`Interbancarias`	Classe que estende a classe `Transferencia`. Responde a operações que permitem realizar transferências que acontecem entre contas interbancárias.

O modelo de classes dos serviços encontra-se descrito na Tabela 5.7 e representado no Anexo F.2 – Modelo de classes dos serviços. Convém notar que algumas das classes dos serviços são equivalentes às classes da aplicação móvel, uma vez que é necessário representar os mesmos conceitos nestes dois ambientes. Por exemplo, a classe `Conta` existe nos dois modelos e representa o mesmo conceito nesses dois ambientes.

Tabela 5.7: Descrição do modelo de classes dos serviços.

Classe	Descrição
SoftwareBanco	Classe responsável por receber um pedido e reencaminhá-lo para a classe que o vai processar.
ISoftwareBanco	Interface que indica quais os serviços que devem ser implementados.
HandlerAutenticação	Classe responsável por agrupar, receber e encaminhar todas as operações que estejam associadas a pedidos de autenticação.
HandlerCliente	Classe responsável por agregar, receber e encaminhar todas as operações que estejam associadas a pedidos relacionados com o cliente.
HandlerTransferencia	Classe responsável por agregar, receber e encaminhar todas as operações que estejam relacionadas com pedidos de transferências.
HandlerConta	Classe responsável por agregar, receber e encaminhar todas as operações que estejam relacionadas com pedidos de contas.
Autenticacao	Classe associada às operações que envolvem dados para o processo de autenticação do cliente.
CatalogoTransferencia	Classe responsável por operações associadas às transferências.
CatalogoConta	Classe responsável por operações associadas às contas.
CatalogoCliente	Classe responsável por operações associadas ao cliente.
Codigo	Classe que representa um código de transferência nos serviços.
Transferencias	Classe que representa uma transferência nos serviços.
Conta	Classe que representa uma conta nos serviços.
Cliente	Classe que representa um cliente nos serviços.
SQLServerDB	Classe responsável pela comunicação com a base de dados. Apenas esta classe tem permissões de acesso aos dados do sistema. Contém diversas funções que permitem efetuar um conjunto de operações na base de dados.
CriaFicheiro	Classe responsável por criar um ficheiro pdf (extensão `.pdf`) com os campos de uma transação bancária e da conta de origem após a conclusão da transferência. Este documento será posteriormente enviado por email.
Seguranca	Classe responsável por garantir a segurança com o exterior dos serviços. Sempre que é necessário existir a transferência de dados para o exterior é utilizada a classe para encriptar os dados.
RSA	Classe que desencripta e encripta dados utilizando a chave privada e pública respetivamente.

Classe	Descrição
SHA256	Classe que utiliza o algoritmo de cifra simétrica (SHA256) para encriptar e desencriptar os dados transmitidos pela rede.
SoftwareGestorBanco	Classe responsável por aceder a outros serviços, de modo a facilitar uma transferência interbancária.
EnviaEmail	Classe que disponibiliza três tipos de envios de emails segundo uma prioridade (por ordem crescente de prioridade): • Envia um email com os dados da transferência bancária, tendo prioridade 3. • Envia um email com um código de transferência previamente criado, tendo prioridade 2. • Envia um email com uma autenticação duvidosa com prioridade 1.
EnviaSMS	Classe responsável por enviar SMS através do *modem* GSM. É a única classe que comunica com o *modem* e que consegue ter essa funcionalidade (enviar SMS).

5.3.2 Desenho da base de dados

Com o avanço exponencial da capacidade computacional, as empresas tiveram a necessidade de armazenar os seus dados em sistemas de *software*, permitindo que o acesso se tornasse mais rápido. Estes *softwares* permitem gerir e assegurar a integridade dos dados empresariais mantendo as relações cliente/empresa. Neste projeto, o *software* empregue foi o *Microsoft SQL Server*. Contudo, independentemente da tecnologia utilizada, existiu a necessidade de realizar uma estrutura lógica da base de dados.

Na Figura 5.4 encontra-se a estrutura desenvolvida e implementada na base de dados utilizada no projeto e na Tabela 5.8 encontra-se uma descrição de cada elemento representado no diagrama.

Tabela 5.8: Descrição da arquitetura da base de dados.

Entidade	Descrição
Cliente	Tabela que representa um utilizador na base de dados e é identificado através do seu id (*primary key*). O campo `nAdesao`, `salt`, `mail` e `telemovel` devem ser únicos, não existindo campos repetidos em diferentes linhas.
Conta	Tabela que representa uma conta, sendo o identificador o `id` (*primary key*). Tanto o `nConta` como o `iban` devem ser únicos, pois são o que identificam e diferenciam as contas.
Transferencia	Tabela que guarda as transferências que são realizadas no sistema. É identificada pelo `id`, id do cliente que realizou a transferência e pela conta de origem da transferência (*primary key*). Possui id do cliente e id da conta como chaves estrangeiras. O tipo identifica o tipo de transferência, podendo conter "I" para transferências interbancárias e "E" para transferências entre contas do mesmo banco.
Conta_Cliente	Tabela que estabelece a relação entre um cliente e uma conta. Identificada pelo seu `id`, id do cliente e da conta (*primary key*). Tem associado o id do cliente e da conta como chaves estrangeiras.

Entidade	Descrição
`Transferencia_Code`	Tabela que guarda os dados de uma transferência a ser realizada por código. É identificada pelo seu `id` e pelas chaves estrangeiras (id do cliente e da conta). Difere da tabela `Transferencia`, pois cada linha nesta tabela não representa uma transação realizada no sistema, mas sim uma possível transferência a ser realizada.
`Movimento`	Tabela que regista de forma permanente todos os movimentos de uma conta (depósitos ou levantamentos). Corresponde a uma entidade fraca, apenas existindo caso exista uma conta. É identificado através do seu `id` e do id da conta correspondente. O tipo identifica o tipo de movimentação, podendo conter "D" para depósitos e "L" para levantamentos.

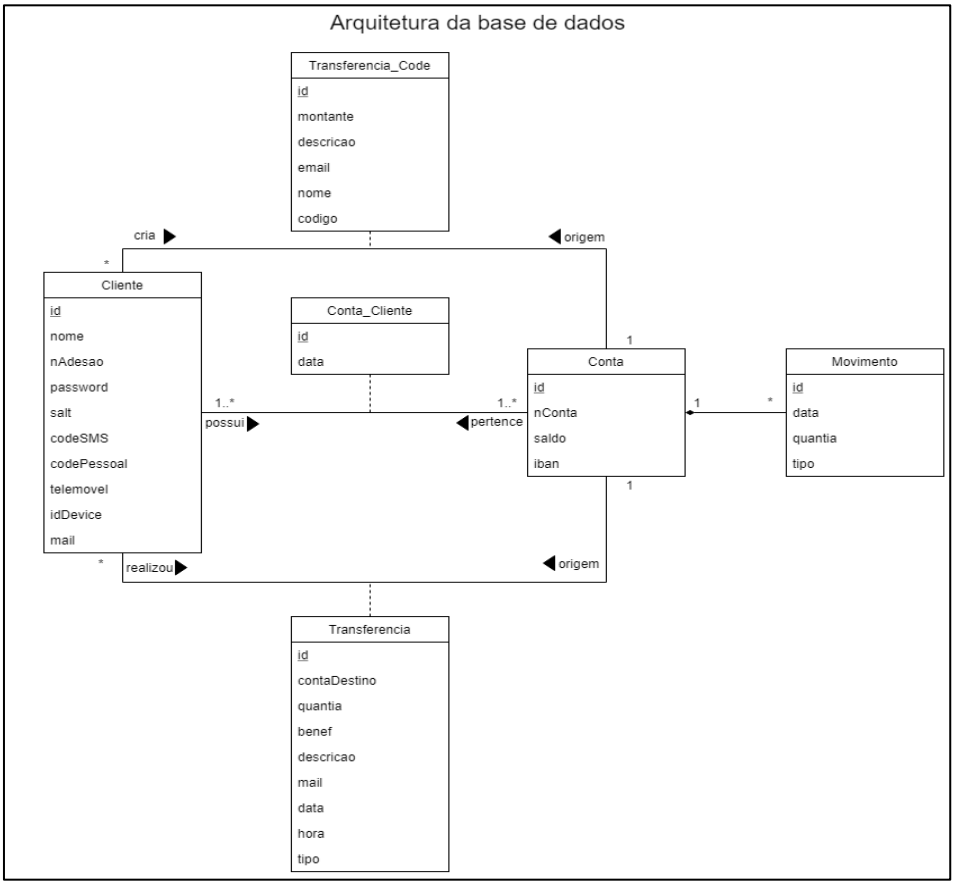


Figura 5.4: Arquitetura da base de dados.
(https://gitlab.com/Henry15/Imagens_da_tese/blob/master/Arquitetura_da_base_de_dados/Arquitetura_da_base_de_dados.png).

5.3.3 Arquitetura de *Software*

Nesta secção vão ser apresentados diferentes tipos de arquiteturas que foram utilizadas e elaboradas neste projeto. Simplificaram o modelo e o modo como os diferentes componentes se interligam e comunicam, tanto a nível de *software* como de *hardware*. A realização de três arquiteturas deveu-se à possibilidade de visualizar pontos de vista diferentes do sistema que permitem ter uma visão orientada a características diferentes do problema e da solução

desenvolvida. Estas várias arquiteturas podem ser estruturadas como versões incrementalmente mais detalhadas e técnicas da totalidade do sistema. A arquitetura em camadas explora a visibilidade dos componentes face ao utilizador, descrevendo numa linguagem de alto nível o tipo de interação do utilizador com cada componente e o fluxo de dados entre os utilizadores e as bases de dados do *back-end*. A arquitetura física tem por base o *hardware* físico a que os componentes estão associados. Por fim, a arquitetura lógica expõe a forma como os vários componentes da solução se relacionam e interagem uns com os outros.

5.3.3.1 Arquitetura em camadas do sistema

A arquitetura em camadas pode ser definida como um processo de decomposição de sistemas complexos em camadas para facilitar a compreensão, implementação, testes e manutenção destes sistemas. De modo a facilitar esta etapa do desenvolvimento, a arquitetura em camadas tornou-se muito importante, pois o desenvolvimento *mobile* está a conquistar cada vez mais relevância no mercado, permitindo que os sistemas tenham diversas interfaces. Na Figura 5.5 encontra-se a arquitetura em camadas do sistema utilizada neste projeto.

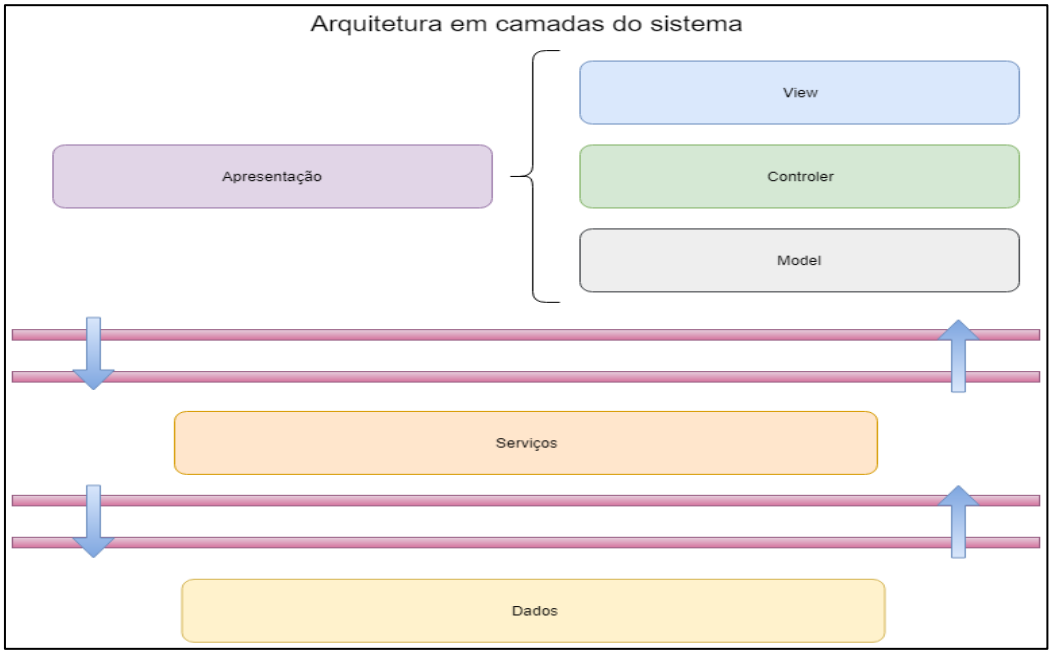


Figura 5.5: Arquitetura em camadas do sistema.
(https://gitlab.com/Henry15/Imagens_da_tese/blob/master/Arquitetura_do_sistema/Camadas/Arquitetura_em_camadas.png).

A arquitetura empregue no projeto é uma arquitetura típica em camadas (N-tier), onde cada uma possui responsabilidades específicas no funcionamento do sistema, limitando a sua interação ao mínimo indispensável, fazendo com que se tornem modulares, ou seja, independentes umas das outras. Cada camada interage com a camada adjacente, não existindo dificuldade na substituição de camadas por outras, desde que as interfaces (contratos) se mantenham.

O sistema encontra-se dividido em três camadas (3-tier):

- **Camada de apresentação –** Corresponde à aplicação móvel, componente do sistema que vai utilizar os serviços. Suporta o padrão de arquitetura de *software* MVC, onde existe uma separação entre a representação da informação e da interação do utilizador.
- **Camada de serviços –** Camada onde se encontram os serviços. Permite que os clientes, por meio da aplicação, tenham acesso aos dados. Faz acesso aos dados por meio de um serviço *web*, realizando uma ponte entre a camada de apresentação e os dados.
- **Camada de dados –** Camada responsável por aceder aos dados e disponibilizá-los para a camada de serviços, sendo responsável pela persistência deles.

5.3.3.2 Camada física do sistema

A Figura 5.6 representa a arquitetura física do sistema e permite destacar o conjunto de elementos de *hardware*, as suas ligações e a forma como os utilizadores interagem com os elementos que pertencem ao sistema.

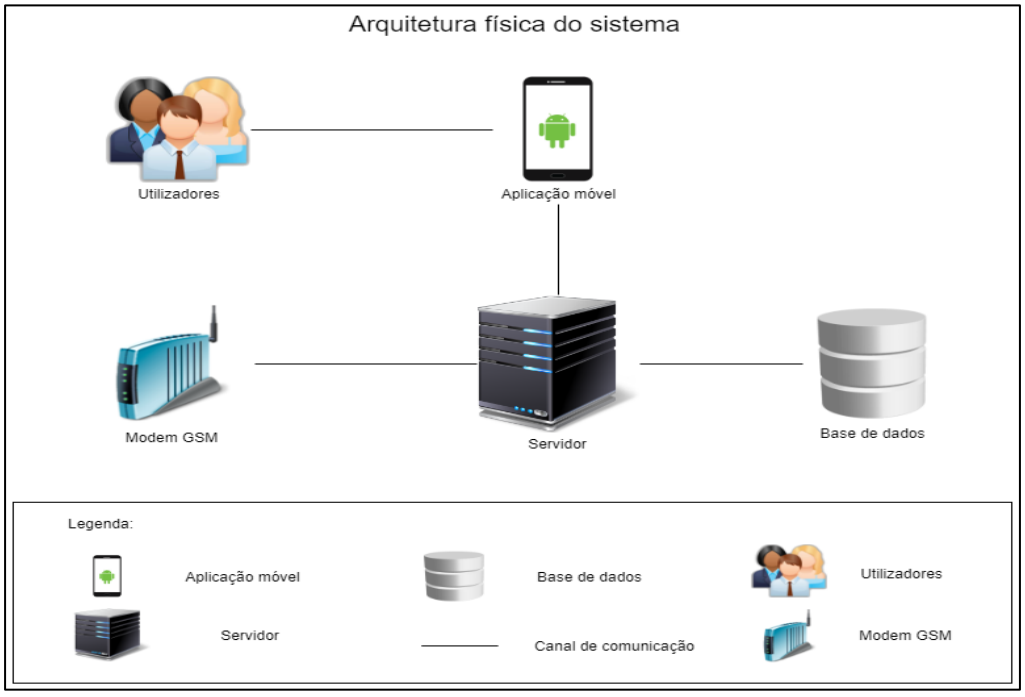


Figura 5.6: Arquitetura física do sistema.
(https://gitlab.com/Henry15/Imagens_da_tese/blob/master/Arquitetura_do_sistema/Fisica/Arquitetura_fisica_do_sistema.png).

A arquitetura física do sistema engloba os seguintes componentes:

- **Utilizadores –** Os utilizadores são o público-alvo e desejam que o sistema satisfaça as suas necessidades no sector bancário.

- **Aplicação móvel –** Porta de entrada para os utilizadores usufruírem do sistema através da interface. Disponibiliza as diversas funcionalidades e permite fazer a gestão de sessões.
- **Servidor –** Máquina que implementa os serviços e permite o acesso aos dados e a outras funções relevantes para a concretização de tarefas.
- **Base de dados –** Componente que armazena e gere os dados dos utilizadores.
- ***Modem* GSM –** Elemento que permite enviar um SMS para um número previamente indicado.

5.3.3.3 Arquitetura lógica do sistema

Uma arquitetura lógica representa a forma como os componentes lógicos de uma solução estão organizados e interligados. Este tipo de arquitetura permite ajudar a entender a comunicação entre os módulos e as classes que os constituem.

A arquitetura lógica do sistema engloba os seguintes módulos:

- **`view` –** Parte lógica responsável pelas interfaces da aplicação móvel.
- **`controller` (aplicação) –** Zona responsável para receber os eventos iniciados pelo utilizador quando interage com a interface móvel.
- **`model` –** Reúne a lógica de negócio. Todas as classes necessárias para criar um desenvolvimento orientado a objetos encontram-se no `model`.
- **`adapter` –** Permite auxiliar as classes que controlam as ações do utilizador. São classes que estendem diversas bibliotecas do *Android* e que permitem obter informação sobre as ações do utilizador.
- **`uteis` (aplicação) –** Módulo com classes que influenciam na lógica da aplicação. Encontram-se classes que estendem a biblioteca disponível pelo *Android* para detetar impressões digitais. Encontram-se também neste módulo diversas classes que ajudam na lógica da aplicação.
- **`segurança` (aplicação) –** Responsável pelas funções de criptografia na aplicação móvel. Modifica os dados que são recebidos e enviados de e para a rede.
- **`services` (aplicação) –** Zona onde se encontra a classe responsável por comunicar com os serviços, de modo a obter os dados necessários ao normal funcionamento da aplicação.
- **`com` –** Porta de entrada para os serviços. Responsável no acesso aos serviços disponíveis.
- **`interfaces` –** Responsável por disponibilizar interfaces para a criação dos serviços implementados.
- **`controller` (serviços) –** Módulo que permite controlar e diferenciar cada pedido. Faz com que os pedidos relacionados com o cliente se encontrem separados de pedidos de transferências, de contas e de autenticação.

- **`business` –** Zona *core* dos serviços. Módulo responsável por responder e preparar a resposta para o cliente que efetuou o pedido para um determinado serviço.
- **`uteis` (serviços) –** Módulo responsável por ajudar em determinadas tarefas de alguns pedidos, nomeadamente a criação de ficheiros de comprovativos de transferência.
- **`segurança` (serviços) –** Módulo que faz a encriptação e desencriptação dos dados dos serviços.
- **`services` (serviços) –** Módulo que comunica com outros serviços, permitindo enviar emails e mensagens de texto por telemóvel.
- **`database` –** Módulo que comunica com a base de dados. Único que tem acesso aos dados a serem utilizados pelos serviços e posteriormente pela aplicação móvel e utilizadores.

No Anexo G (G.1) encontra-se a arquitetura lógica do sistema organizado de acordo com a arquitetura em camadas na secção 5.3.3.1. No Anexo G encontram--se também a arquitetura lógica da aplicação móvel (G.2) e dos serviços (G.3) respetivamente, de uma forma detalhada, apresentado as classes que formam cada módulo bem como parte das suas relações.

5.4 *Layouts*

Nesta fase desenvolveu-se a criação e elaboração de protótipos de interface da aplicação móvel, tornando-se possível a resolução de alguns problemas encontrados na fase de levantamento de requisitos, nomeadamente o excesso de interfaces para a realização de tarefas.

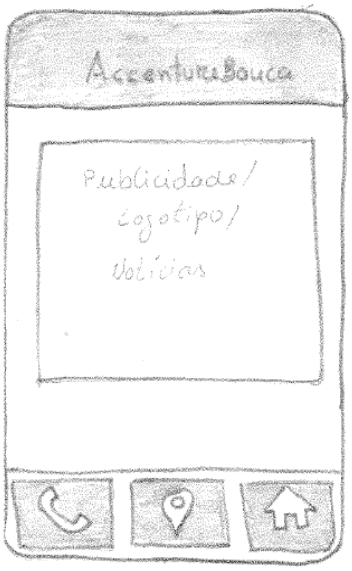
Esta secção encontra-se dividida em três partes (**protótipos de baixa fidelidade**, **protótipos de alta fidelidade** e **fluxo de navegação**) que propõem uma interface para a aplicação, as quais culimaram, por fim, na interface de produção.

5.4.1 Protótipos de baixa fidelidade

Estes protótipos permitem uma pré-visualização gráfica da interface da aplicação, dando a possibilidade de ajuste e alteração não requerendo muito tempo nem esforço. Foram elaborados protótipos de baixa fidelidade para todas as interfaces essenciais deste projeto, com o intuito de ter uma perspetiva inicial do que poderia vir a ser a interface. Estes protótipos são importantes para o desenvolvimento dos *layouts*, uma vez que se adequam às necessidades e objetivos pretendidos, não levando muito tempo a serem desenvolvidos e alterados.

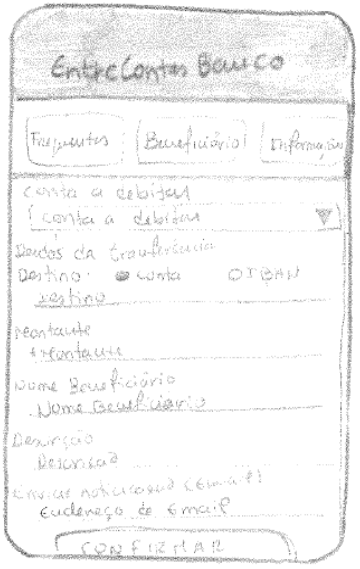
Uma vez que a participação do cliente foi cancelada, os protótipos foram desenvolvidos de acordo com os problemas encontrados na aplicação anterior (já existente) e nos objetivos a nível de interfaces. Para isso, construiu-se um fluxo de navegação o mais simples possível, não requerendo um grande número de *layouts* para a realização de tarefas.

Em baixo na Figura 5.7 encontram-se alguns protótipos de baixa fidelidade desenvolvidos nesta fase.



a) Ecrã inicial.

b) Ecrã de login.



c) Ecrã de transferências entre contas do mesmo banco.

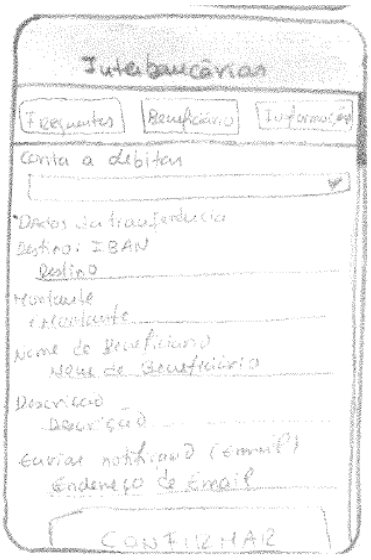
d) Ecrã de autenticação de uma tarefa.

e) Ecrã *home*.

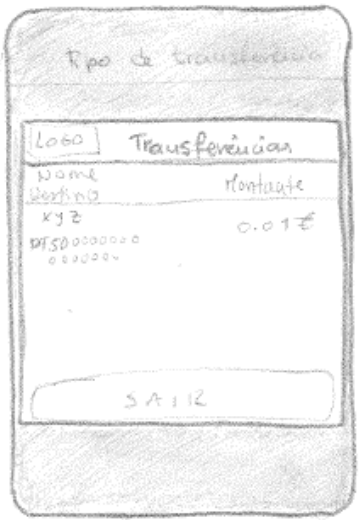
f) Ecrã da lista de transferências ordenadas por ordem de frequência.

g) Ecrã do menu de navegação da aplicação.

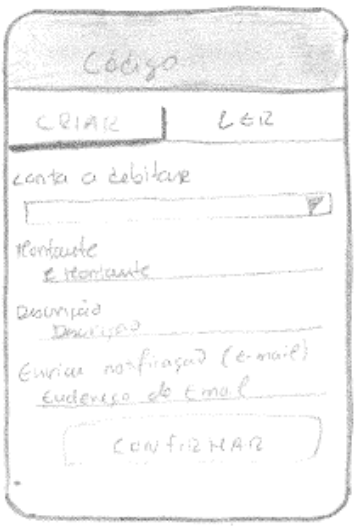
h) Ecrã do menu de transferências.



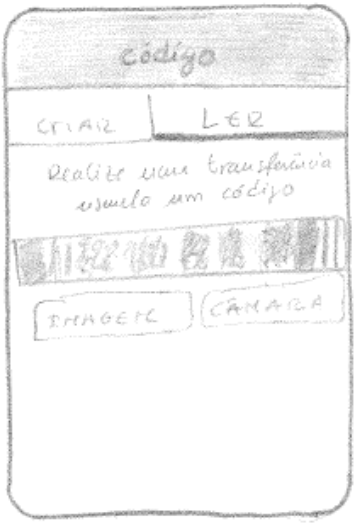
i) Ecrã de transferências interbancárias.



j) Ecrã da lista de transferências ordenadas por ordem de beneficiário.

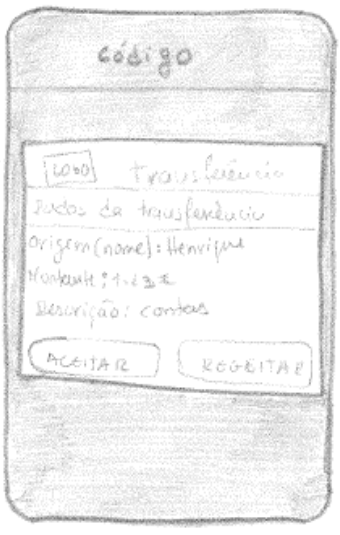


k) Ecrã para criar um código de transferência.



l) Ecrã para ler um código de transferência.

m) Ecrã de um código de transferência criado.



n) Ecrã dos dados associados, após leitura de um código de transferência.

Figura 5.7: Protótipos de baixa fidelidade.

5.4.2 Protótipos de alta fidelidade

A produção dos protótipos de alta fidelidade visa aproximar as interfaces desenvolvidas nos protótipos de baixa fidelidade com os *layouts* finais da aplicação. Permite a simulação do fluxo completo de todas as funcionalidades, de modo a avaliar a interação do utilizador com um *template* próximo do produto final. Estes protótipos necessitaram de mais tempo e recursos para serem construídos, sendo que a sua fase de produção ocorreu numa etapa mais avançada no desenvolvimento do produto.

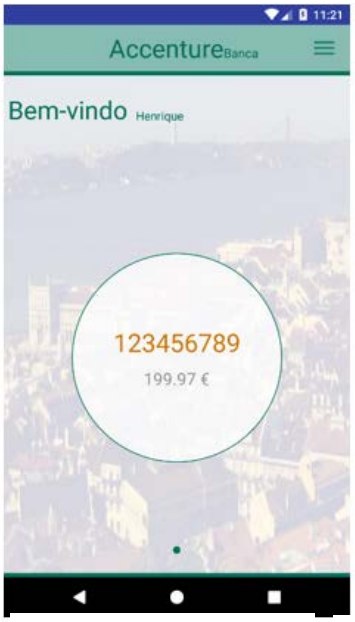
Em baixo na Figura 5.8 seguem-se alguns protótipos de alta fidelidade que foram desenvolvidos e implementados no decorrer do projeto.

a) Ecrã inicial.

b) Ecrã de login.

c) Ecrã de transferências entre contas do mesmo banco.

d) Ecrã de autenticação de uma tarefa.



e) Ecrã *home*.

f) Ecrã da lista de transferências ordenadas por ordem de frequência.

g) Ecrã do menu de navegação da aplicação.

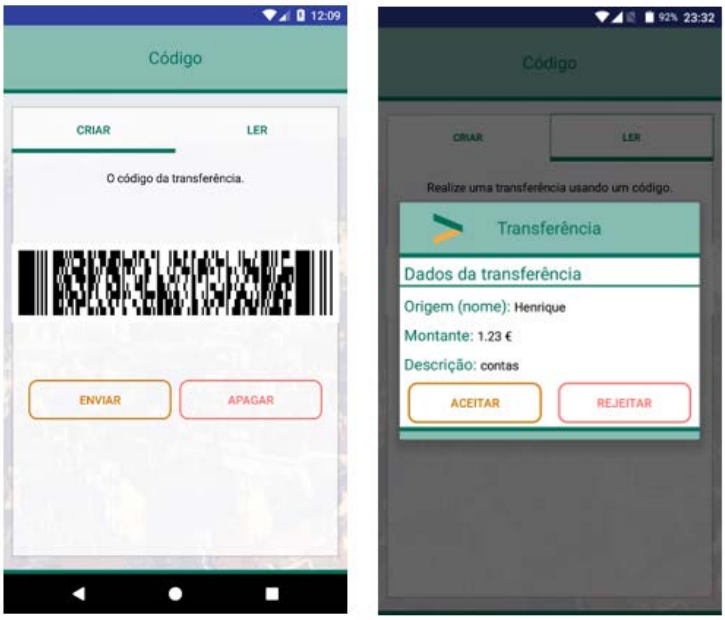
h) Ecrã do menu de transferências.

i) Ecrã de transferências interbancárias.

j) Ecrã da lista de transferências ordenadas por ordem de beneficiário.

k) Ecrã para criar um código de transferência.

l) Ecrã para ler um código de transferência.



m) Ecrã de um código de transferência criado.

n) Ecrã dos dados associados, após leitura de um código de transferência.

Figura 5.8: Protótipos de alta fidelidade.

5.4.3 Fluxo de navegação

O fluxo de navegação corresponde a uma etapa importante do desenvolvimento de sistemas para avaliar e prever as sequências de navegação pelas quais os utilizadores irão passar.

No levantamento de requisitos foi elaborado este tipo de diagramas com o intuito de verificar toda a navegação que a aplicação anterior (já existente) suportava. Do mesmo modo, foram realizados diagramas de fluxo de navegação com o propósito de se tornar claro o fluxo de navegação dos utilizadores. Nas Figura 5.9, Figura 5.10, Figura 5.11 e Figura 5.12 representam-se, respetivamente, os diagramas de controlo de fluxo de transferências entre contas do mesmo banco, interbancárias e por código (criação e leitura). A realização de controlo de fluxo apenas para as tarefas de transferência deveu-se ao facto de este projeto ter um foco especial nesse tipo de tarefas, sendo estas as tarefas que serão mais utilizadas pelos utilizadores. A principal diferença entre os diagramas apresentados surge relativamente à funcionalidade e, por isso, o fluxo é específico e único para cada. O desenvolvimento dos diagramas de fluxo permitiu uma melhor visualização da comunicação entre as diversas interfaces da aplicação móvel. A criação da sua estrutura foi elaborada através dos problemas encontrados na secção 5.2.1.1.

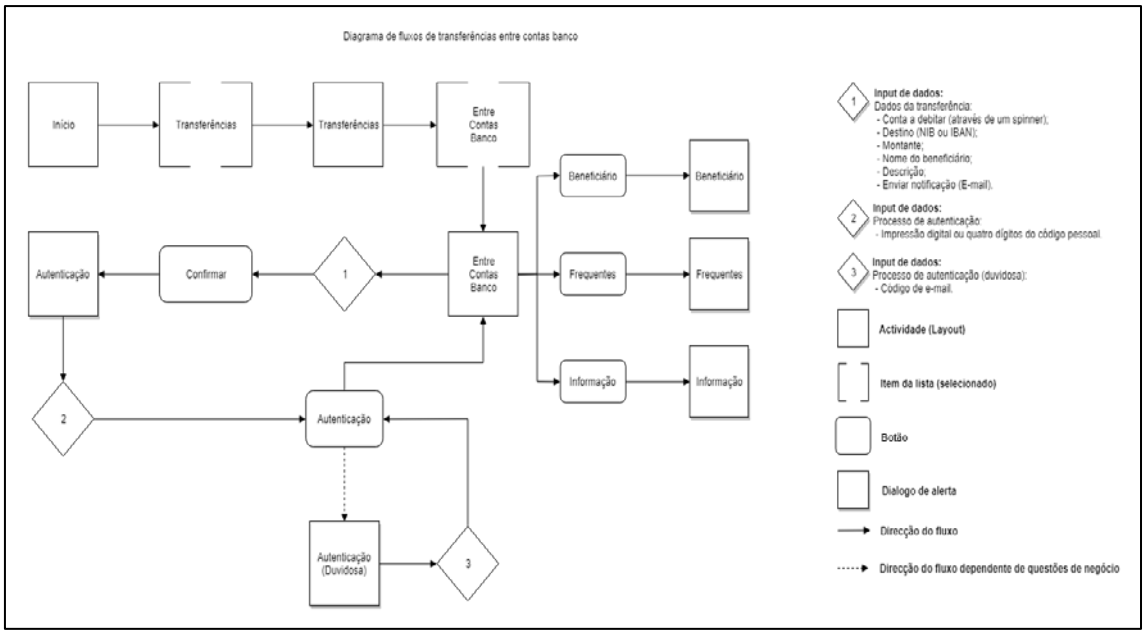


Figura 5.9: Diagrama de controlo de fluxo de transferências entre contas do mesmo banco.

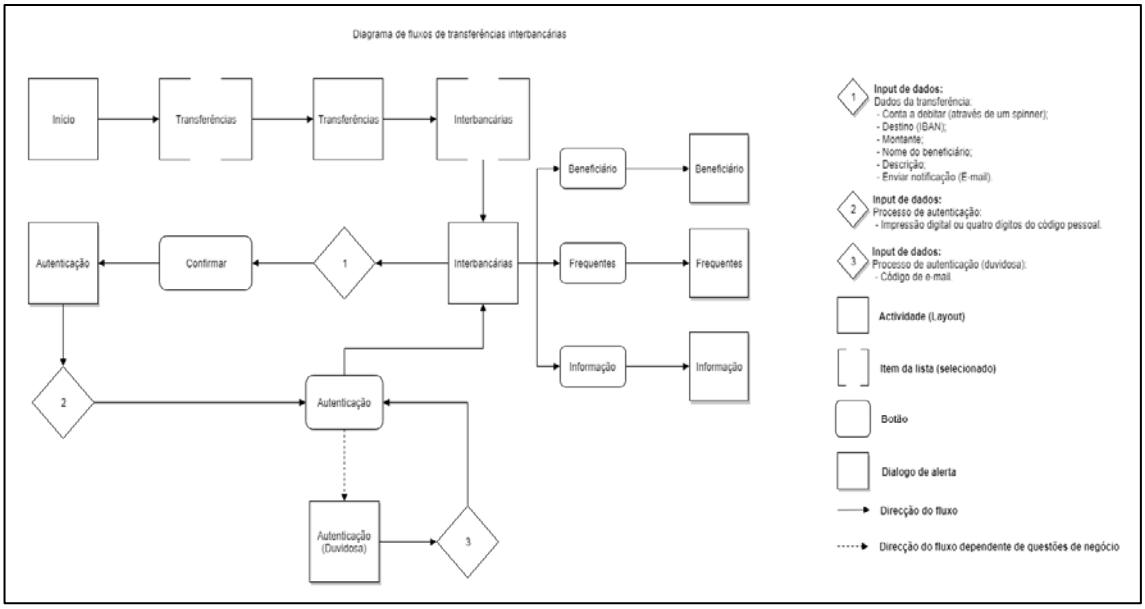


Figura 5.10: Diagrama de controlo de fluxo de transferências entre contas interbancárias.

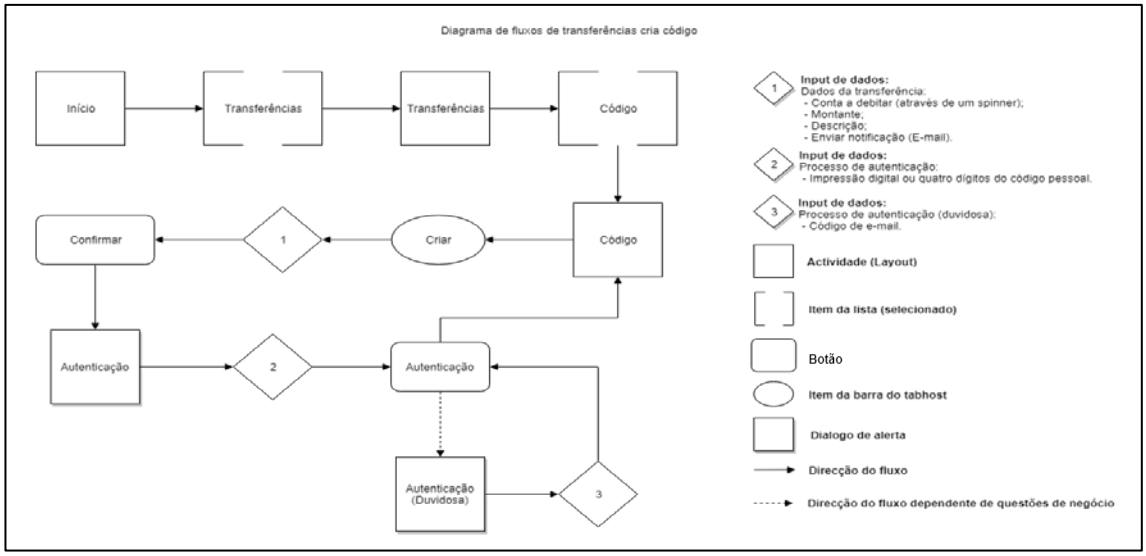


Figura 5.11: Diagrama de controlo de fluxo de transferências por código (criação de código).

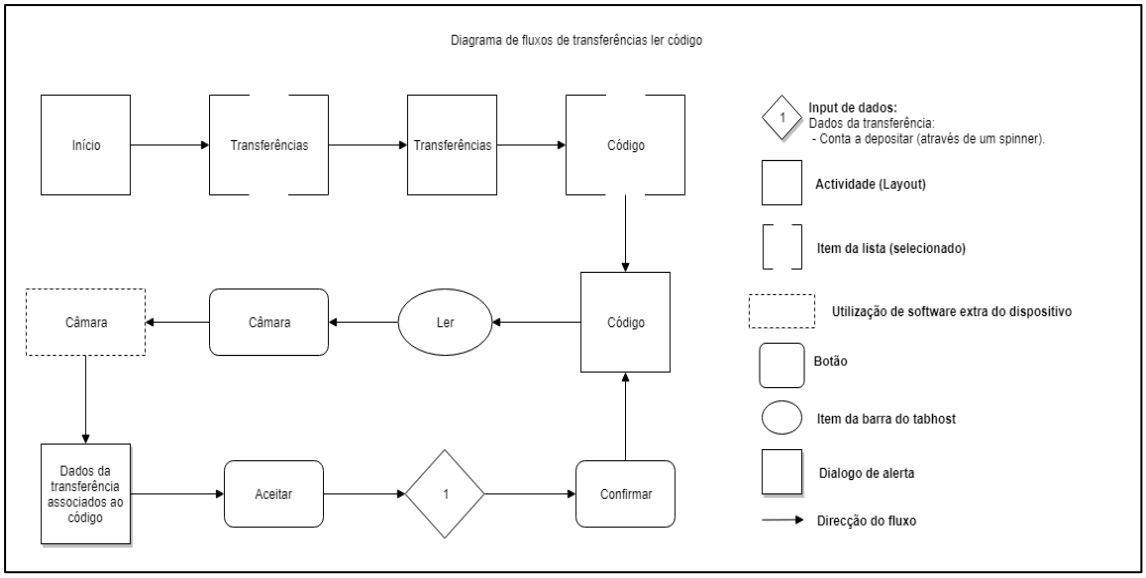


Figura 5.12: Diagrama de controlo de fluxo de transferências por código (leitura de código).

Comparação com a aplicação existente

A parte inicial do fluxo das tarefas não foi alterada, uma vez que mudanças drásticas podiam afetar negativamente os utilizadores. A alteração focou-se no número de atividades (*layouts*) e de processos de autenticação. Com o objetivo de diminuir o número de interfaces utilizou-se o *widget* (*spinner*) para mostrar as contas associadas a um determinado utilizador e fazer com que o fluxo se mantenha na mesma atividade. Todos os processos de autenticação foram alterados para diálogos de alertas (`AlertDialog`), fazendo com que o utilizador não se perca no fluxo de interação.

A aplicação existente e disponibilizada pelo cliente possuía demasiados processos de autenticação para a realização de tarefas chave. Neste projeto, a diminuição do número de processos de autenticação e a facilitação do fluxo foi um objetivo alcançado pela comparação dos diagramas de fluxo da aplicação já existente e da aplicação móvel desenvolvida neste projeto.

O sistema desenvolvido agrupa dois processos de autenticação, sendo que um deles pode ser realizado de duas formas (quatro dígitos do código pessoal ou impressão digital). Apesar de possuir dois processos, apenas um deles vai ser utilizado constantemente, sendo que o outro irá funcionar como controlo para identificar autenticações duvidosas. O processo de autenticação mais utilizado e considerado principal é o que permite os dois métodos de autenticação: um por meio de impressão digital e outro através de quatro dígitos do código pessoal de cada utilizador. Este processo foi desenhado para minimizar a possibilidade de autenticações erradas. Ainda assim, caso o sistema considere que exista uma autenticação indevida, este vai encaminhar o utilizador para o segundo processo de autenticação (autenticação duvidosa). Como a probabilidade de um utilizador passar pela autenticação duvidosa é reduzida, o tempo despendido e a complexidade de fluxo são diminutos na realização de funcionalidades chave comparativamente com a aplicação anterior.

5.5 Implementação

Para que a aplicação desenvolvida tenha impacto junto dos consumidores finais, é essencial que a fase de implementação siga o desenho e requisitos analisados, estruturados e criados nas fases anteriores. Corresponde a uma etapa importante no ciclo de vida de um projeto, dado que é nesta fase que se começa a ter a maior interação e perceção do produto final.

Nesta secção vai ser descrito o processo de desenvolvimento e *frameworks* utilizadas para a implementação dos diversos componentes que constituem o sistema.

5.5.1 Base de dados

Uma base de dados corresponde a um simples repositório de informação relacionado e estruturado, consoante a sua finalidade. Permite consultar, atualizar, inserir e apagar dados de modo a gerir toda a informação armazenada. A base de dados implementada no projeto seguiu a arquitetura da base de dados desenvolvida na secção 5.3.3, permitindo estruturar os dados de uma forma lógica, de modo a que o acesso aos dados fosse o mais eficiente possível.

A *framework* utilizada para armazenar e gerir os dados do sistema foi o *Microsoft SQL Server*, baseada num sistema relacional (SIBD), fazendo uso da linguagem SQL (*Structured Query Language*) para armazenamento e recuperação de dados solicitados por aplicações de *software* (secção 3.4.1).

De modo a implementar o desenho estrutural da base de dados desenvolvido na secção 5.3, e uma vez que foi utilizada uma base relacional, foi necessário o uso da linguagem SQL para proceder à criação de tabelas e às suas relações. A linguagem SQL é uma linguagem padrão para a gestão de dados que interagem com as principais bases de dados relacionais. Foi necessária a implementação da camada de dados com o objetivo de tornar a aplicação móvel o mais real possível, possibilitando assim a realização de transferências mesmo não acedendo aos dados do cliente. Tornou-se essencial para verificar as rápidas alterações que a lógica do negócio efetuou nos dados durante a realização de tarefas.

5.5.2 Serviços *web*

A utilização de serviços *web* no projeto permite que várias aplicações e clientes diferentes comuniquem, utilizando os protocolos de padrão da Internet. Facilita a comunicação de componentes que são desenvolvidos separadamente e possivelmente com linguagens diferentes. Para tal é necessária a utilização de uma linguagem universal (XML e JSON, por exemplo).

Para facilitar a utilização de dados por parte da aplicação cliente foi desenvolvido no âmbito do projeto um conjunto de serviços *web* que permitem a ponte entre a camada de apresentação e dos dados a serem consumidos pela aplicação móvel. Os serviços *web* implementados seguem uma arquitetura REST, onde cada serviço possui um URL específico, bem como um conjunto de parâmetros de entrada. Foram ainda desenvolvidos segundo o protocolo *stateless* (sem estado), onde o tratamento dos pedidos é independente entre si.

Neste projeto foi criado um *web dynamic project* no IDE do *Eclipse*, uma vez que permite a criação de projetos com orientação a serviços. Faz a estruturação e configuração de todo o projeto, facilitando a fase de implementação. É uma aplicação baseada na *Apache Axis2* através de um ficheiro `.jar` descrita anteriormente na secção 3.4.1.

5.5.2.1 REST

O REST define-se como um estilo arquitetural que permite a ligação entre máquinas através do protocolo HTTP e tem como objetivo a construção de aplicações em rede. Das vantagens destacam-se: facilidade de implementação e compreensão, diminuição da latência, escalável para um grande número de clientes e é mais leve para os sistemas do que o SOAP. As desvantagens são as seguintes: não abrange todos os padrões dos serviços *web* e ausência de um padrão comum para descrever formalmente um serviço REST. Por estas razões, escolheu-se o REST em vez do SOAP (Costello, 2018).

5.5.2.2 Acesso aos dados

Com o objetivo de se aceder aos dados armazenados na base de dados foi essencial existir um processo de autenticação, que permitisse o acesso devido, de modo a poderem ser utilizados no processamento de um serviço.

A classe `SQLServerDB` possui um método que faz a conexão com a base de dados e só tem permissão para realizar operações aos dados caso tenha sido dada essa autorização na autenticação. Corresponde à classe que tem exclusividade no acesso aos dados. Após a autenticação, esta realiza operações de leitura e escrita na base de dados, passando os dados recolhidos na operação devidamente estruturados à camada de processamento de pedidos.

5.5.2.3 Resposta a pedidos

Para facilitar o tratamento de uma resposta a um pedido, os dados que são recebidos da base de dados estão estruturados em classes, que são analisados e posteriormente retornados ao cliente.

As respostas são enviadas em formato JSON, fazendo-se uso da biblioteca `Gson` para efetuar essa passagem de formato. O `Gson` é uma biblioteca *Java* que permite a conversão de objetos *Java* numa representação JSON, através da função `toJson()`. Em baixo segue-se um exemplo de resposta a um pedido.

```
public String getCliente(
      @PathParam("nAdesao") String nAdesao,
      @PathParam("password") String password,
      @PathParam("chave") String chave
) {
      Cliente retorno = handlerCliente.getCliente(
                        nAdesao, password, chave);
      Gson gson = new Gson();
      return gson.toJson(retorno);
}
```

5.5.2.4 Envio de um SMS através de um *modem* GSM

Um dos requisitos que o sistema devia suportar era o de permitir enviar um SMS (RF10), de acordo com a Tabela 5.1). Após a verificação de várias opções para realizar este requisito, concluiu-se que a melhor solução seria o uso de um *modem* GSM para o envio de SMS, já explicado na secção 3.4.2.3.

Existiu um maior trabalho e tempo despendido para adquirir conhecimento sobre o funcionamento desta tecnologia, porque o conhecimento era limitado e a informação disponibilizada sobre como é feita a comunicação com um *modem* GSM em programação era escassa.

O *modem* GSM permite a introdução de um cartão SIM e faz a ponte de ligação com uma determinada operadora telefónica. Comporta-se como um telemóvel, dando a possibilidade de serem enviadas SMS e serem efetuadas ligações à Internet. Relativamente ao envio de SMS por parte do *modem* GSM, este tem de ser configurado para permitir o envio de um SMS e necessita

de sete segundos para enviar uma mensagem de texto, ficando indisponível para o sistema durante esse período. Toda a comunicação é feita utilizando uma porta serial e é processada fora da resposta a um pedido. De modo a que seja concluído um SMS para ser enviado é necessário digitar na porta serial o comando "`control+z`", que corresponde ao conjunto de caracteres em *Java* de `\u001a`. Em baixo segue o código para o envio de um SMS usando um *modem* GSM.

```
private static CommPortIdentifier portId;
private static SerialPort serialPort;
private static OutputStream outputStream;

public static void enviaCodigoSMS(String codigo, String numero)
throws Exception {
      // identificação da porta onde o modem está ligado
      portId = CommPortIdentifier.getPortIdentifier("COM4");
      Random rand = new Random();
      int x = rand.nextInt();
      serialPort = (SerialPort) portId.open(
            "Teste" + Integer.toString(x), 1000);
      serialPort.notifyOnDataAvailable(true);
      serialPort.setSerialPortParams(9600,
                  SerialPort.DATABITS_8,
                  SerialPort.STOPBITS_1,
                  SerialPort.PARITY_NONE);
      outputStream = serialPort.getOutputStream();
      enviaModoTexto(codigo, numero);
}

private static void enviaModoTexto(String codigo, String numero)
throws Exception{
      // Mete o modem em modo texto
      String x1 = "AT+CMGF=1\r\n";
      outputStream.write(x1.getBytes());
      Thread.sleep(1000);

      // Indica o número de destino
      String numeroNovo = "+351"+numero;
      String x2 = "AT+CMGS=\""+numeroNovo+"\",145\r\n";
      outputStream.write(x2.getBytes());
      Thread.sleep(1000);

      // Envia a mensagem que deseja
      String codigoNovo = meteEspacosCode(codigo);
      String x3 = "Em baixo segue o seu codigo sms:\n"
            +codigoNovo
            +"\nGuarde-o num local seguro.\nObrigado."
            +"\u001a";
      outputStream.write(x3.getBytes());
      Thread.sleep(500);

      outputStream.close();
      serialPort.close();
}
```

5.5.2.5 Envio de emails

Em *software*, o tempo de resposta do sistema é muito importante para os utilizadores. Deste modo, o tempo que os serviços levam a processar cada pedido é crucial.

Durante a fase de implementação, reparou-se que o envio de um email (autenticação duvidosa, código de transferência e comprovativo de transferência) não podia ocorrer no meio da resposta a um pedido. Isto levaria a que o tempo de resposta aumentasse consideravelmente. A solução passou por criar uma *task* (iniciada com o servidor) responsável por enviar emails. A *task* criada tem associada uma fila de espera onde são postos todos os emails que devem ser enviados. Sempre que um pedido necessite de enviar um email, este vai colocá-lo na fila de espera para ser enviado, continuando o processamento do pedido, não aguardando que o email seja enviado.

A fila de espera que está associada à *task* atribui uma determinada prioridade aos tipos de emails a serem enviados. Essa prioridade encontra-se descrita em baixo:

- **Prioridade 1** – nível de maior relevância, onde se encontram os emails associados às autenticações duvidosas;
- **Prioridade 2** – nível de relevância intermédio, que corresponde a *tasks* de envio de emails com código de transferência;
- **Prioridade 3** – nível de menor relevância, atribuído aos emails onde são enviados os comprovativos de transferência bancária.

A atribuição destas prioridades segue uma lógica que dá maior relevância às tarefas em que o utilizador quer aceder à aplicação ou efetuar transações, havendo a necessidade de se autenticar em ambos os casos. Deixando para menos prioritário o envio de comprovativos de transações. Sempre que um email com uma determinada prioridade *n* chega à fila de espera, este passa automaticamente à frente de todos os emails com prioridade menor que *n*.

5.5.3 Aplicação móvel

Neste projeto, houve a necessidade de separar componentes uma vez que correspondia a uma aplicação móvel e, deste modo facilitava o desenvolvimento e testes. Foi utilizado o *Android Studio* como *framework* de desenvolvimento *mobile*. Cria um ambiente de desenvolvimento integrado (IDE), o que possibilita o desenvolvimento e testes rápidos (usando AVD) de aplicações para a plataforma *Android*.

5.5.3.1 MVC

A aplicação móvel corresponde à camada de apresentação. De modo a estruturar os módulos e classes elaborados, foi seguido o padrão de arquitetura MVC (*Model-View-Controller*) adquirido na Licenciatura em Engenharia Informática. O MVC faz a separação de

uma aplicação em três partes interconectadas. Foi feito para separar a informação da interação que o utilizador realiza, aumentando a eficiência da implementação e permitindo o desenvolvimento em paralelo. É composto por modelo (`model`), visão (`view`) e controlador (`controller`). O modelo agrupa os dados da aplicação, regras de negócio, lógica e funções *core* da aplicação. A visão é representada por qualquer saída de dados (tabela, diagrama, campos textuais) para o ecrã do dispositivo móvel, sendo possível existirem diversas visões dos mesmos dados a apresentar. O controlador permite a ligação entre a visão e o modelo, detetando ações e convertendo-as em comandos tanto para o modelo como para a visão (Barros *et al*, 2007).

5.5.3.2 Criação de pedidos

Para que o utilizador consiga dispor dos dados para efetuar as funcionalidades que a aplicação móvel disponibiliza, é crucial que esta aceda à informação. Visto que o único componente que possui permissões de acesso aos dados é a camada de dados, a aplicação efetua pedidos aos serviços, que consultam a camada da base de dados, para realizar as suas funcionalidades.

Uma das boas formas de programação de uma aplicação *Android* ao aceder a serviços é o facto de não ser possível as chamadas ocorrerem na UI *Thread* da aplicação, sendo necessário criar uma *thread* auxiliar para o realizar, informando o utilizador de que algo está a acontecer, não indicando que o sistema parou.

Neste projeto foi utilizada a classe `AsyncTask`, que permite a criação de *threads* em *background*, possibilitando a realização de outras tarefas sem interferir com o pedido. A classe disponibiliza um conjunto de métodos que ajudam na preparação e receção dos pedidos.

Para se efetuar a conexão aos serviços foi utilizada a biblioteca `HttpURLConnection`, sendo necessária a utilização do URL associado ao serviço específico, o tipo de requisição e o *timeout*.

5.5.3.3 Receção e tratamento de respostas

Os dados que chegam dos serviços são em formato JSON e é necessário fazer a transcrição do formato JSON para uma estrutura idêntica na aplicação móvel. Do mesmo modo que foi utilizada a biblioteca `Gson` para passar de uma estrutura para JSON, a mesma biblioteca permite o processo inverso. Desta forma, foi invocado o método `fromJson()`facilitando a passagem de dados entre os componentes.

A estrutura JSON é recebida na classe `AsyncTask` no método `onPostExecute(String s)`, encaminhado o resultado para o controlador (`controller`) que por sua vez vai notificar e passar o resultado ao modelo (`model`). No modelo, existem classes responsáveis por passar de JSON para uma estrutura de dados, sendo

que essa classe diverge de funcionalidade para funcionalidade. Em baixo apresenta-se um exemplo do tratamento de uma resposta a um pedido.

```
public Cliente getClienteResposta(String resposta) {

    if(resposta.equals("servico")){
        return new Cliente();
    }else{
        Gson gson = new Gson();

        Cliente cliente = null;
        Type c = new TypeToken<Cliente>() {}.getType();
        cliente = gson.fromJson(resposta, c);

        if(cliente.getNome() == null){
            return null;
        }

        …
    }
}
```

5.6 Testes

Na atualidade, o mercado *mobile* tem grande impacto nas tarefas da população mundial e por isso, o correto funcionamento das aplicações nos dispositivos é de grande relevância. De modo a garantir a qualidade do *software* é necessária a realização de testes rigorosos ao longo do ciclo de vida do projeto.

A construção de *software* está sujeita a diversos tipos de problemas e alterações, que podem culminar num produto diferente daquele que se esperava durante a criação de requisitos. Para evitar este problema, é necessário criar e executar testes de *software* para que possam ser detetados erros antes de o produto ser lançado no mercado. Nesta fase de projeto, o objetivo foi detetar todas as falhas que o sistema possuía, de forma a serem corrigidas em interações posteriores (Delamaro *et al*. 2016).

Esta secção descreve todos os testes que foram realizados de forma a garantir a qualidade do produto desenvolvido neste projeto.

5.6.1 Testes de sistema

Com todos os componentes do sistema implementados e integrados, efetuaram-se um conjunto de testes de modo a verificar se o sistema funcionava corretamente como um todo. Com este tipo de testes, a finalidade foi a interação com o sistema de forma a realizar todas as funcionalidades, analisando eventuais falhas ou incoerências.

A fim de possibilitar este tipo de testes foi necessário implementar os serviços *web* num servidor localmente, preparar a aplicação móvel para utilizar os serviços e garantir que tanto os

serviços como a aplicação móvel estariam ligados à mesma rede Internet. Os testes de sistema foram realizados de forma manual. No entanto, existe a possibilidade de realização deste tipo de testes de forma automática através das ferramentas: UI *Automator, Espresso, Mockito*.

Teste 1 – Inicialização e acesso à conta

A realização deste teste permitiu verificar a inicialização da aplicação. Uma vez que a aplicação móvel utiliza serviços é necessário que o dispositivo *hardware* esteja conectado à Internet. Durante esta tarefa conseguimos distinguir um utilizador que tenta aceder à sua conta pela primeira vez, de um utilizador que já passou por essa fase. Só será necessário efetuar a escolha do código pessoal e receber o envio de um SMS no primeiro acesso à conta, fazendo com que todos os futuros acessos necessitem apenas da autenticação do utilizador (código pessoal ou impressão digital). Como a aplicação faz conexão com os serviços de forma segura, sempre que a aplicação é iniciada, esta vai obter a chave pública do servidor antes de mostrar a interface inicial.

Após esta fase inicial, o utilizador consegue usufruir de todas as funcionalidades implementadas na aplicação. Podemos satisfazer vários requisitos com a realização deste teste, nomeadamente: efetuar o login com um número de adesão e uma password; envio de um SMS com um código e autenticação utilizando um código pessoal ou por meios biométricos. Os *screenshots* do Anexo H.1 apresentam os ecrãs visualizados pelo utilizador durante esta tarefa.

Teste 2 – Realização de uma transferência entre contas do mesmo banco

A realização deste teste apenas é possível caso o utilizador consiga passar o teste anterior. Trata-se de um teste que permite a realização de uma transferência entre contas do mesmo banco. Para que consiga realizar a operação, o utilizador deve seguir os seguintes passos:

- Abrir o menu lateral;
- Selecionar “TRANSFERÊNCIAS”;
- Selecionar “ENTRE CONTAS BANCO”;
- Efetuar o correto preenchimento dos campos;
- Realizar a autenticação.

De modo a agilizar este processo é possível visualizar as transferências que cada utilizador efetuou para contas do mesmo banco. Para tal são visíveis as transferências por ordem de frequência ou por ordem de beneficiário. Caso o utilizador necessite de realizar uma transferência, basta visualizar as transferências já realizadas e selecionar a pretendida. Os *screenshots* do Anexo H.2 apresentam os ecrãs visualizados pelo utilizador durante a realização do teste 2.

Com a realização deste teste podemos satisfazer vários requisitos descritos na fase de análise: realização de transferências entre contas do mesmo banco; visualização de transferências frequentes e beneficiário já realizadas; envio do comprovativo da transferência por email e autenticação utilizando um código pessoal ou impressão digital.

Teste 3 – Realização de uma transferência interbancária

Neste teste, o objetivo foi testar se a aplicação permite a realização de transferências entre contas de bancos diferentes. Para que tal aconteça é necessário que o sistema comunique com uma entidade que faça a gestão dos bancos. Essa comunicação é transparente ao utilizador, pois este apenas preenche os campos necessários, efetua a autenticação e aguarda que o sistema realize a transação.

Para que consiga realizar a operação, o utilizador deve seguir os seguintes passos:

- Abrir o menu lateral;
- Selecionar “TRANSFERÊNCIAS”;
- Selecionar “INTERBANCÁRIAS”;
- Efetuar o correto preenchimento dos campos;
- Realizar a autenticação.

Para facilitar a realização de transferências, é possível, neste tipo de operação, visualizar o conjunto de transações realizadas para outras contas com origem em bancos diferentes. Através dos botões “FREQUENTES” e “BENEFICIÁRIO” é possível visualizar as transferências que já foram realizadas. Seguindo a mesma lógica, torna-se também possível a repetição de uma transação que se encontra registada no sistema (que já foi anteriormente efetuada). Para tal, o utilizador pode encontrar a transferência que necessita de realizar através dos botões acima citados, selecionar a que pretende e efetuar o processo de autenticação. Os *screenshots* do Anexo H.3 apresentam os ecrãs visualizados pelo utilizador durante a realização do teste 3.

Com a realização deste teste podemos satisfazer vários requisitos descritos na fase de análise: realização de uma transferência interbancária; visualização de transferências frequentes e beneficiário já realizadas; envio do comprovativo da transferência por email e autenticação utilizando um código pessoal ou impressão digital.

Teste 4 – Realização de um código de transferência

Era intenção da *Accenture* tornar a aplicação mais inovadora. Posto isto foi implementada uma nova forma de realizar uma transação preenchendo menos campos e não existindo a necessidade de um conhecimento prévio da conta de destino por parte do utilizador.

Neste caso, o utilizador apenas tem a necessidade de indicar a conta de origem, o montante, uma descrição da transação e o seu email, caso tenha interesse em ser notificado após o término da transação.

Este teste permitiu verificar se o sistema efetua a criação de um código que permita realizar uma transferência. Para a realização deste teste é necessário que o utilizador siga os seguintes passos:

- Abrir o menu lateral;
- Selecionar "TRANSFERÊNCIAS";
- Selecionar "CÓDIGO";
- Efetuar o correto preenchimento dos campos;
- Realizar a autenticação.

Com a criação do código, o utilizador tem a possibilidade de mostrar o código para ser lido através do processo de *scan*, enviar o código para o email do representante da conta de destino ou apagar o código criado. Para que seja enviado o código para um email, é necessário que o utilizador indique qual o endereço de email de destino. Para realizar esta tarefa é necessário que sejam realizados os seguintes passos: indicar que pretende enviar o código previamente criado e indicar o email de destino. Os *screenshots* do Anexo H.4 apresentam os ecrãs visualizados pelo utilizador durante a realização do teste 4.

Outra ação que o utilizador tem e que correspondia a um requisito é o facto de ser possível apagar um código já criado. Deste modo, apenas necessita de indicar que quer apagar o código previamente criado e que ainda não tenha sido enviado ou utilizado pelo utilizador recetor do código de transferência.

Nestes testes podemos verificar que são cumpridos os requisitos definidos no levantamento de requisitos: cria um código que permite realizar uma transferência; efetua o envio por email do código de transferência; apaga o código de transferência e realização de uma autenticação utilizando um código pessoal ou impressão digital.

Teste 5 – Leitura e realização de uma transferência por código

É intuito neste teste a validação da leitura de um código de transferência, bem como a sua aceitação. Um utilizador tem a possibilidade de ler um código de duas maneiras diferentes: através de uma imagem (efetuando o *download* para o seu dispositivo móvel) ou através do s*can* (alinhando a câmara com o código a ler). Para a realização desta tarefa é necessário efetuar os seguintes passos:

- Abrir o menu lateral;

- Selecionar “TRANSFERÊNCIAS”;
- Selecionar “CÓDIGO”;
- Clicar na *tab* “LER”;
- Indicar o modo como deseja realizar a leitura do código.

Após a leitura, o utilizador tem a possibilidade de aceitar ou rejeitar a transferência. Caso aceite, irá realizar-se uma transferência onde o utilizador apenas tem de indicar uma das suas contas para depositar o dinheiro. No caso de rejeitar, esta apaga o código de todo o sistema (aplicação móvel e *back-end*). Os *screenshots* do Anexo H.5 apresentam os ecrãs visualizados pelo utilizador durante a realização do teste 5.

Com este teste podemos validar os requisitos: efetuar a leitura de um código de transferências; realizar uma transferência por código e apagar um código de transferência do sistema.

5.6.2 Testes de carga

Os serviços são a principal porta de acesso aos dados que vão ser utilizados pela aplicação cliente e posteriormente apresentados nos ecrãs dos dispositivos *hardware*. Desta forma, tornam-se fundamentais os testes de carga aos serviços para garantir uma boa resposta por parte do *software* implementado. Para a realização destes testes foi utilizada a *framework SoapUI*, facilitando assim o conjunto de testes a realizar aos serviços REST. Para a preparação de testes definiram-se os serviços a serem testados e os objetivos a serem alcançados. Uma vez que os serviços não tinham acesso aos dados do cliente foi necessária a simulação de contas para garantir a fiabilidade da lógica dos serviços, caso estes venham a ter acesso aos dados do cliente bancário.

Os **serviços** a serem testados são:

- **Teste 1** – Serviço para se obter a chave pública do servidor;
- **Teste 2** – Serviço para se obter um cliente através do número de adesão e da password;
- **Teste 3** – Serviço para obter as contas associadas ao cliente;
- **Teste 4** – Serviço para realizar uma transferência entre contas do mesmo banco;
- **Teste 5** – Serviço para realizar uma transferência interbancária;
- **Teste 6** – Serviço para criar um código de transferência;
- **Teste 7** – Serviço para leitura de um código de transferência.

Foi considerado como **objetivo** a atingir o “Tempo médio de resposta a pedidos” de 3 segundos.

Todos os testes foram realizados em vagas, com um tempo de espera entre elas pré-definido. As configurações aplicadas aos serviços foram:

- **Utilizadores** – em cada vaga de testes foram lançados 1000 pedidos aos serviços *web*;
- ***Test delay*** – As vagas foram espaçadas com 6 segundos entre elas;
- **Tempo de duração** – foram lançadas vagas durante 10 minutos, o que significa que foram lançadas 100 vagas;
- ***Warm-up*** – porque existe uma fase de "aquecimento" do serviço, em que os resultados dos testes de carga são superiores aos atingidos quando os serviços já estão a funcionar há muito tempo, foram descartados os resultados atingidos no minuto 1, ou seja, só foram usados os resultados das últimas 90 vagas.

Na Tabela 5.9 apresentam-se os resultados para os testes de cada serviço bem como os dados relevantes obtidos na sua execução.

Tabela 5.9: Resultados dos testes realizados.

Resultados dos testes		
Teste	**Total de testes**	**Tempo médio de resposta**
Teste 1	21365	0,02 segundos
Teste 2	20986	0,13 segundos
Teste 3	21112	0,10 segundos
Teste 4	17549	1,33 segundos
Teste 5	15595	2,22 segundos
Teste 6	20945	0,14 segundos
Teste 7	20997	0,13 segundos

Através da observação dos dados dos testes, verifica-se que os dados obtidos correspondem aos objetivos estipulados. Uma boa fase de implementação dos serviços facilitou bastante a realização dos testes.

Os serviços, que necessitam de mais tempo de processamento médio (teste 4 e 5), são serviços que realizam muitas leituras e escritas na base de dados e possuem algum processamento, o que limita a sua eficiência. Por outro lado, uma vez que o envio de emails e SMS é feito fora do processamento de um pedido, tornam as respostas muito mais rápidas.

Em suma, os testes efetuados ao sistema e serviços garantem uma boa resposta à realização de tarefas. Apenas foram encontradas algumas falhas na interface (falta do símbolo euro (€) nas representações monetárias e incoerência nos títulos dos diálogos relativamente às tarefas associadas). Para melhorar a resposta aos serviços deveria ser implementado um sistema de

cache, uma vez que iriam diminuir o número de acessos à base de dados. Com o mesmo objetivo, podemos aumentar o número de servidores que disponibilizam os serviços, fazendo com o tempo médio de espera de um pedido a ser aceite diminua.

5.7 Rede

Como já foi referido anteriormente com a anulação de determinadas fases do ciclo de desenvolvimento, houve alteração nas infraestruturas do projeto. Inicialmente eram para ser utilizadas máquinas pertencentes ao cliente, mas com esta alteração, houve a necessidade de utilizar os serviços e base de dados localmente.

Para se poderem efetuar testes de sistema foi necessário criar uma rede interna, uma vez que os serviços estavam implementados numa máquina com IP privado, o que leva a que apenas dispositivos da mesma rede consigam aceder aos serviços (ver secção 3.5).

Existem duas possibilidades para a criação de uma rede interna: através de um *router* ou de um computador. Quando o *router* é ligado, cria uma rede automaticamente, uma vez que já vem configurado de fábrica. Por outro lado, o computador deve ser configurado para se tornar um *router* na rede. Neste projeto utilizou-se a configuração de um computador como *router*, fazendo a utilização do programa *Connectify*. O *Connectify* permite a criação de uma rede, sendo necessário ligar o computador à Internet, indicar qual a rede que queremos partilhar, o nome e a password da rede a criar e iniciá-la. Após estes passos o computador passou a ser visto na rede como um *router* e a rede criada ficou disponível para ser acedida.

A Figura 5.13 representa a rede criada para efetuar testes de sistema e a Tabela 5.10 apresenta uma descrição dos componentes da rede.

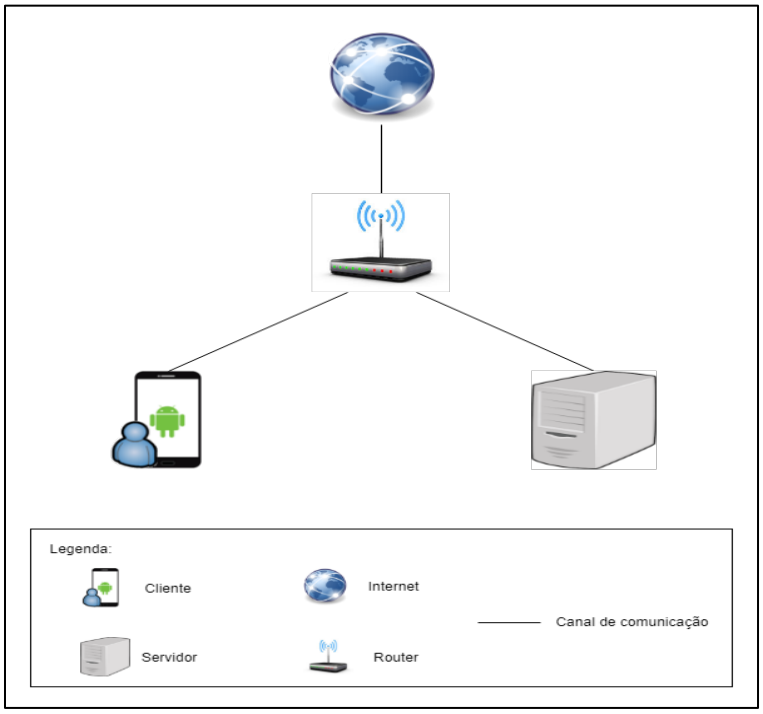


Figura 5.13: Rede criada para efetuar testes de sistema.

Tabela 5.10: Descrição dos elementos da rede.

Elemento	Descrição
Cliente (dispositivo móvel)	Dispositivo *hardware* ligado à rede interna que o *router* criou. É-lhe atribuído um IP privado.
Servidor (serviços + base de dados)	Servidor que disponibiliza os serviços. Está ligado à mesma rede interna que os clientes (rede privada). Possui igualmente um IP privado.
Router	Único elemento com IP público, que contém uma rede interna e por isso faz utilização do protocolo NAT. Permite que os elementos da rede interna tenham acesso ao exterior.
Internet	Máquinas ligadas à rede mundial.

Capítulo 6

Conclusões

6.1 Considerações finais

As instituições do sector bancário investem em serviços direcionados para o cliente com recurso a dispositivos móveis. Entres eles, o *mobile banking* é o mais recentemente utilizado, pois o uso desta tecnologia possibilita uma maior facilidade de interação com o banco por parte dos clientes, facilitando o seu dia-a-dia nas tarefas bancárias. Em particular, note-se que em Portugal a utilização de dispositivos móveis é superior a 1 *per capita*, fator que contribui para que o *mobile banking* tenha condições para crescer exponencialmente (Mallat *et al*., 2004; Laukkanen e Cruz, 2009).

A participação neste projeto proporcionou o conhecimento teórico e prático de novas metodologias empregues e o planeamento de trabalho foi cumprido como planeado inicialmente (secção 2.3).

Um dos objetivos principais da *Accenture* para este projeto foi a utilização de tecnologias e métodos inovadores para a realização de tarefas, de modo a melhorar a usabilidade de toda a aplicação *mobile*. Para tal utilizaram-se as seguintes tecnologias e métodos: a impressão digital como método de autenticação e a realização de uma transação entre contas do mesmo banco através de um código de barras.

A limitação mais importante deste trabalho foi o facto de o cliente ter suspendido algumas etapas do projeto e com isso, de se ter perdido um pouco o contacto com um ambiente empresarial e com o trabalho em equipa. Além disso, como a aplicação móvel final não foi disponibilizada para o mercado, não existem dados estatísticos nem comentários sobre a sua aceitação pelo público-alvo.

Apesar de já não existir uma comunicação com o banco cliente, todo o projeto foi desenvolvido com base nos problemas da aplicação já existente, com o intuito de a melhorar. Deste modo, a aplicação desenvolvida durante este projeto teria impacto positivo junto dos clientes do banco. Uma vez que tornou a aplicação móvel mais tecnológica (realização de uma transferência através de um código de barras), simplificou o processo de autenticação (utilização

da impressão digital) e do fluxo de navegação, garantindo que o sistema não limitasse o acesso a serviços.

Uma aplicação móvel é sempre uma mais valia para qualquer negócio, uma vez que fideliza os clientes, desde que as aplicações cumpram com o prometido. Para tal é necessário não limitarem o acesso às funcionalidades e acompanharem o avanço tecnológico.

Os utilizadores de aplicações móveis procuram facilidade na realização de tarefas e deste modo, a concretização de uma transferência por código de barras facilita a sua realização. Permite que qualquer utilizador realize uma transação não sendo necessário o conhecimento do NIB ou IBAN, fazendo com que se torne mais agradável a realização de uma transferência.

A aplicação desenvolvida permite que seja usada por qualquer banco. Desta forma é necessário ter em atenção: o tipo de interface (as cores e disposição dos elementos na interface), as políticas de segurança e a arquitetura, pois cada instituição bancária possui políticas de segurança e arquiteturas próprias nos seus sistemas de *software*.

Como as aplicações estão em constante alteração, existiu uma preocupação durante este projeto de as mesmas poderem ser realizadas posteriormente por outra equipa de desenvolvimento. Deste modo, tanto o código da aplicação móvel, como o código dos serviços estão armazenados num repositório *git*, facilitando assim o processo de controlo de versões e de futuras atualizações, além de haver documentação (parcialmente incluída nos anexos deste relatório) e comentários explicativos ao longo do código.

6.2 Perspetivas futuras

Atualmente a tecnologia está permanentemente em evolução e atualização. Desta forma, é essencial que as aplicações acompanhem essa evolução. A ***Accenture*** procura encontrar soluções para os seus clientes no âmbito ***technology*** e ***digital*** de modo a satisfazer as suas necessidades com as tecnologias mais recentemente desenvolvidas.

Como o projeto desenvolvido não é um ciclo fechado, apresentam-se de seguida algumas melhorias que poderão ser desenvolvidas em fases posteriores deste projeto:

- Desenvolvimento de um sistema de notificações para a realização de transações concluídas. Nomeadamente, quando um utilizador realiza uma transferência por um código de barras, o sistema iria enviar uma notificação para o dispositivo do utilizador da conta de origem, a informar que a transferência teria sido realizada com sucesso.
- Desenvolvimento de transações SEPA (*Single Euro Payments Area).* A SEPA é uma área única de pagamentos em euros, que abrange consumidores, empresas e administração pública de 33 países, facilitando o movimento transacional, uma vez

que possibilita as transações de uma forma mais simples e eficaz, reduzindo tempo e custos.

- Realização de transações interbancárias através de um *barcode* (código de barras). De modo a realizar este tipo de transferência, é necessário que exista mais do que um banco que permita a realização de transferências por código de barras. Por outro lado, é necessário perceber onde faz sentido armazenar os dados de transferência (conta origem, montante, descrição) associados a um código de barras: se no sistema do banco de origem da transferência; se no sistema do banco de destino da transferência ou se no sistema do regulador das entidades bancárias.

Referências bibliográficas

Accenture (2016). Fintech and the evolving landscape: landing points for the industry. 1-12.

Accenture (2018). Accenture Portugal distinguida como Top Employer 2018 pelo terceiro ano consecutivo. Portugal Newsroom, 15 de fevereiro de 2018.

Alves, W. P. (2017). Desenvolvendo aplicações com Xamarin. Novatec, 1-269.

Apache Software Foundation (2018). Welcome to Apache Axis2/Java. https://axis.apache.org/axis2/java/core. 2018.

Barros, T., Silva, M. e Espínola, E. (2007). State MVC: Estendendo o padrão MVC para uso no desenvolvimento de aplicações para dispositivos móveis. CESAR.

Carvalho, N. F. (2013). A Banca Portuguesa - O desenvolvimento da banca portuguesa e as instituições incorporadas pelo BNU e pela CGD. Gabinete do Património Histórico da CGD, setembro de 2013.

Choudhury, A. (2017). The Oldest Operational Banks in The World. Economics, Wordatlas, 25 de abril de 2017.

Comer, D. E. (2016). Redes de Computadores e Internet. 6ª Edição. Bookman. Purdue University.

Costa, C. (2016). Banco de Portugal: País tem 5 agências bancárias por 10 mil habitantes, mas se calhar ainda é muito. Diário de Notícias, 24 de novembro de 2016.

Costello, R. L. (2018). Building web services the REST way. Xfront. Tutorials and Articles on XML and Web Tecnologies.

Delamaro, M. E., Maldonado, J. C. e Jino, M. (2016). Introdução ao teste de software. 2ª Edição. Elsevier.

Developer Android (2018). https://developer.android.com/. 2018.

Eclipse Foundation (2018). The Platform for Open Innovation and Collaboration. https://www.eclipse.org. 2018.

Elidol, M. e Erol, C. S. (2014). Mobile Application Development with Agile Methodology. Economics and Business Communication Challenges: International Week, 72-86.

Faustino, G., Calazans, H. e Lima, W. (2017). Android e a influência do Sistema Operacional Linux. Tecnologias em Projeção. Vol. 8. 100-111.

Fedoseev, G., Degtyarey, A., Iakushkin, O., *et al*. (2016). A continuous integration system for MPD Root: Deployment and setup in GitLab. Saint-Petersburg State University. 525-529.

Ferreira, L. M. (2018). Altice e Huawei reforçam parceria para implementar 5G em Portugal. Jornal Económico, 26 de fevereiro de 2018.

Foster, E. C. e Godbole, S. (2016). Overview of Microsoft SQL Server. Database Systems. Apress, Berkeley, CA, 461-467.

Gok N. e Khanna, N. (2013). Building Hybrid Android Apps with Java and JavaScript. Applying Native Device APIs. O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

Gueron, S. e Krasnov, V. (2012). Simultaneous hashing of multiple messages. J. Information Security, 3(4), 319-325.

Hildenbrand, J. (2017). How does Android save your fingerprints? And how secure is it? Android Central, ASK AC, 26 de setembro de 2017.

Kruchten, P. (2004). The Rational Unified Process: an introduction – 3rd. ed. Addison-Wesley, US.

Larman, C. e Basili, V. R. (2003). Iterative and Incremental Development: A Brief History. IEEE Computer Society. Vol. 6, 47-56.

Laukkanen, T. e Cruz, P. (2009). Comparing Consumer Resistance to Mobile Banking in Finland and Portugal. Communications in Computer and Information Science, Vol. 48.

Liao, S., Shao, P. S., Wang, H., *et at*. (1999). The adoption of virtual banking: an empirical study. International Journal of Information Management, Vol. 19 (1), 63-74.

Mallat, N., Rossi, M., Tuunainen, V., *et at*. (2004). Mobile Banking Services. Communications of the ACM, Vol. 47, 42-46.

Marktest (2015). 650 mil utilizadores de mobile banking. Comportamento face aos bancos, Grupo Marktest, 14 de julho de 2015.

Marktest (2017). 2,5 milhões de utilizadores de Internet Banking. Comportamento face aos bancos, Grupo Marktest, 21 de fevereiro de 2017.

Marktest (2018). Android casa vez mais usado em Portugal. Grupo Marktest, 9 de janeiro de 2018.

Martínez-Pérez, B., Lopez-Coronado, M. e Diez, I. T. (2014). Privacy and security in Mobile Health Apps: A review and recommendations. Universidad de Valladoil. Journal of Medical Systems, dezembro de 2014.

Mendes, J. A. (2002). A empresa bancária em Portugal no séc. XX: Evolução e estratégias. Gestão e Desenvolvimento, 11, 39-56.

Morais, J. R. (2017). Banca fecha balcões, mas internet banking não cresce. Diário de Notícias, 20 de fevereiro de 2017.

Oluwatosin, H. S. (2014). Client-Server Model. IOSR, Journal of Computer Engineering. 67-71.

Pandya, V. e Shukla, D. (2012). GSM Modem Based Data Acquisition System. International Journal of Computational Engineering Research. Vol. 2(5), 1662-1667.

Pereira, L. C. e Silva, M. L. (2009). Android para desenvolvedores. Arquiteturas e Desenvolvimento. Brasport. 1-219.

Pinto, P. (2010). Criptografia simétrica e assimétrica. Sabe a diferença? Networking, pplware, 7 de dezembro de 2010.

Pinto, P. (2010). Wireshark 1.26 – O melhor sniffer para redes informáticas. Networking, pplware, 30 de janeiro de 2010.

Pressman, R. S. (2005). Software Engineering: A Practitioner´s Approach – 6th ed. McGraw-Hill series in computer science. 1-860.

Reuters (2017). World´s first ATM machine turns to gold on 50th birthday. Business News, 27 de junho de 2017.

Rocha, J. (2017). Ekonomista: Bancos em Portugal – o guia completo. Contas e bancos, 18 de agosto de 2017.

Rocha, J. (2017). Público: Único concorrente da rede Multibanco já tem 300 caixas em Portugal. Público, Multibanco, 2 de janeiro de 2017.

Salnikov, N. (2017). Most Popular Java Application Servers (2017 Edition). Dzone, Java Zone, Analysis, 30 de maio de 2017.

Santos, A., Kroll, J., Sales, A., *et al*. (2016). Investigating the Adoption of Agile Practices in Mobile Application Development. Proceedings of the 18th International Conference on Enterprise Information Systems, 1, 490-497.

Schwaber, K. (2004). Agile project management with scrum. Microsoft Press Redmond, WA, EUA.

Shaikh, A. e Karjaluoto, H. (2015). Mobile Banking adoption: A literature review. Telematics and Informatics, 32 (1).

Shukla, S., Khare, V., Garg, S., *et at*. (2013). Comparative Study of 1G, 2G, 3G and 4G. Journal of Engineering, Computers & Applied Sciences, Vol. 2(4), 55-63.

SIBS (2018). Forward Payment Solutions – Em ATM: Características e funcionalidades.

Silva, L., Pires, D., Neto, S. (2015). Desenvolvimento de Aplicações para Dispositivos Móveis: Tipos e Exemplo de Aplicação na plataforma iOS. II Workshop de Iniciação Científica em Sistemas de Informação. 25-28.

Sinicki, A. S. (2018). What is Google Fuchsia? Is this the new Android? Android Authority, 22 de março de 2018.

SoapUI (2018). The world´s most widely used open source API testing tool. SoapUI by Smartbear. https://www.soapui.org/open-source.html.

Sommerville, I. (2011). Software Engineering – 9th ed. Addison-Wesley, EUA. 1-767.

Tannam, E. (2018). Huawei boss warns the potencial of 5G is becoming overblown. Siliconrepublic, 18 de abril de 2018.

TEC-IT (2018). PDF417 (2D Barcode). TEC-IT Datenverarbeitung GmbH, Austria.

Twilio (2018). Twilio Client Android SKD. https://www.twilio.com/docs/voice/client/android.

Weiss, S. (2002). Handheld usability. Usable Products Company, New York, 1-267.

Ye, G. *et al*. (2017). Cracking Android Pattern Lock in Five Attempts. NDSS. Internet Society. 2017.

Yuan, C. P., Wahid, N., Kasim, S., *et al*. (2017). UTHM Student Planner. Acta Informatica Malaysia (AIM), 1, 17-21.

Zechner, M., DiMarzio, J. F., Green, R. (2016). First Steps with Android Studio. Beginning Android Games. Apress, Berkeley, CA, 15-32.

Anexos

Anexo A – Descrição dos casos de uso

A.1 – UC1: Login de utilizador

Ator Principal:

Utilizador.

Interesses:

Utilizador: No final do caso de uso, o Utilizador está autenticado na sua conta, para poder ter acesso aos seus dados bancários e efetuar as suas possíveis transferências e ações através da aplicação.

Pré-Condições:

O Utilizador tem de estar registado no sistema. Deve possuir um número de adesão (nove dígitos) e uma password (nove dígitos).

Pós-Condições:

O Utilizador está autenticado perante o sistema e pode ter acesso aos seus dados bancários e efetuar todas as funcionalidades que o sistema disponibiliza.

Cenário Principal de Sucesso:

1. O Utilizador indica que pretende fazer login no sistema.
2. O sistema pede ao Utilizador o seu número de adesão e a sua password.
3. O Utilizador indica o seu número de adesão e a sua password.
4. O sistema verifica se existe um Utilizador registado com o número de adesão e password indicadas pelo Utilizador e identifica o Utilizador. Verifica se possui código SMS e código pessoal.
5. O Utilizador informa o sistema que pretende autenticar-se com o código pessoal.
6. O sistema mostra as quatro posições do código pessoal que o Utilizador tem de digitar.
7. O Utilizador indica os dígitos do código pessoal que estão nas posições pedidas pelo sistema.
8. O sistema compara os ID dos dispositivos e os dígitos que o Utilizador introduziu e confirma a autenticação e login ao Utilizador.

Extensões (Cenário Principal de Sucesso):

3a. O sistema não encontra registado nenhum utilizador com o número de adesão e password indicadas pelo Utilizador.

1. O sistema informa o Utilizador que não existe nenhum utilizador registado com aquele número de adesão e com aquela password e termina o caso de uso.

4a. O sistema verifica que o Utilizador ainda não possui código SMS nem código pessoal.

1. O sistema pede o número de telemóvel ao Utilizador.
2. O Utilizador indica o seu número de telemóvel.
3. O sistema compara o número digitado com o número de telemóvel guardado, cria um código e envia-o através de um SMS para o número digitado.
4. O Utilizador introduz o código recebido e o seu código pessoal.
5. O sistema compara o código SMS introduzido com o enviado e guarda o código pessoal associado ao Utilizador.
6. Continua no passo 5 (Cenário Principal de Sucesso).

Extensões (extensão 4a.):

3a. O número telemóvel digitado pelo Utilizador é diferente do número de telemóvel guardado e associado a este Utilizador.

1. O sistema informa o Utilizador e termina o caso de uso.

5a. O código SMS introduzido é diferente do enviado.

1. O sistema informa o Utilizador e continua no passo 4 (extensão 4a.).

5b. O código pessoal digitado é fraco.

1. O sistema informa o Utilizador e continua no passo 4 (extensão 4a.).

Continuação da extensão cenário principal de sucesso:

7a. O sistema invalida os dígitos indicados pelo Utilizador.

1. O sistema informa o Utilizador e termina o caso de uso.

5a. O Utilizador informa o sistema que pretende autenticar-se com o sensor de impressão digital.

1. Após o sistema operativo autenticar o Utilizador, o sistema compara o ID do dispositivo atual com o armazenado no sistema para verificar se está a ser feita uma autenticação duvidosa.
2. Se não existir nenhum ID do dispositivo armazenado, o sistema vai armazenar o atual, associando-o ao Utilizador.
3. O sistema autentica o Utilizador e continua no passo 8 (Cenário Principal de Sucesso).

Extensões (extensão 5a.):

1a. O sistema está a correr num sistema operativo que não suporta esta funcionalidade (<*Android* 6.0).

1. O sistema informa o Utilizador que não é possível aceder a este método de autenticação e continua no passo 6 (Cenário Principal de Sucesso).

1b. O sistema está a correr num dispositivo *hardware* que não possui sensor de impressão digital.

1. O sistema informa o Utilizador que não é possível aceder a este método de autenticação e continua no passo 6 (Cenário Principal de Sucesso).

1c. O sistema está a correr num dispositivo *hardware* que não possui nenhuma impressão digital registada no sistema operativo.

1. O sistema informa o Utilizador que não tem nenhuma impressão digital registada e continua no passo 6 (Cenário Principal de Sucesso).

1d. Os ID dos dispositivos de *hardware* são diferentes.

1. O sistema informa o Utilizador que existe uma autenticação duvidosa.
2. O sistema cria um código.
3. O sistema envia o código criado para o email do Utilizador.
4. O Utilizador digita o código do email e o sistema compara-o com o que foi enviado para o email do Utilizador.
5. O sistema armazena o novo ID do dispositivo e confirma a autenticação e login ao Utilizador.

Extensões (extensão 5a.1d.):

4a. O código digitado é diferente do código do email.

1. O sistema informa o Utilizador e termina o caso de uso.

Continuação da extensão cenário principal de sucesso:

8a. Os ID dos dispositivos de *hardware* são diferentes.

1. O sistema informa o Utilizador que existe uma autenticação duvidosa.
2. O sistema cria um código.
3. O sistema envia o código para o email do Utilizador.
4. O Utilizador digita o código do email e o sistema compara-o com o que foi enviado para o email do Utilizador.
5. O sistema armazena o novo ID do dispositivo e confirma a autenticação e login ao Utilizador.

Extensões (extensão 8a.):

4a. O código digitado é diferente do código do email.

1. O sistema informa o Utilizador e termina o caso de uso.

A.2 – UC2: Transferência entre contas do mesmo banco

Ator Principal:

Utilizador.

Interesses:

Utilizador: Pretende efetuar uma transferência de uma das suas contas para outra conta associada ao mesmo banco.

Pré-Condições:

O Utilizador está autenticado perante o sistema.

Pós-Condições:

A transferência foi efetuada com sucesso.

Cenário Principal de Sucesso:

1. O Utilizador indica ao sistema que pretende efetuar uma transferência entre contas do mesmo banco.
2. O Utilizador indica ao sistema que pretende ver as suas contas.
3. O sistema mostra as contas associadas ao Utilizador.
4. O Utilizador confirma a transferência preenchendo todos os campos necessários.
5. O Utilizador informa o sistema que pretende autenticar-se com o código pessoal.
6. O sistema mostra as quatro posições do código pessoal que o Utilizador tem de digitar.
7. O Utilizador indica os dígitos do código pessoal que estão nas posições pedidas pelo sistema.
8. O sistema compara os ID e realiza a transferência.
9. O sistema informa o Utilizador que a transferência foi realizada com sucesso.

Extensões (Cenário Principal de Sucesso):

4a. O Utilizador não preenche todos os campos.

1. O sistema informa o Utilizador que deve preencher todos os campos para poder realizar a transferência e o caso de uso termina.

4b. Os campos preenchidos pelo Utilizador estão inválidos.

1. O sistema informa o Utilizador que tem os campos errados, finaliza o caso de uso e pede-lhe que volte a introduzir os dados corretos para realizar a transferência.

8a. O sistema está indisponível para realizar a transferência.

1. O sistema deve informar o Utilizador que não é possível realizar a transferência atualmente e termina o caso de uso.

8b. Os ID dos dispositivos de *hardware* são diferentes.

1. O sistema informa o Utilizador que existe uma autenticação duvidosa.
2. O sistema cria um código.
3. O sistema envia o código criado para o email do Utilizador.
4. O Utilizador digita o código do email e o sistema compara-o com o que foi enviado para o email do Utilizador.
5. O sistema confirma a autenticação do Utilizador, realiza a transferência e continua no passo 9 (Cenário Principal de Sucesso).

Extensões (extensão 8b.):

4a. O código digitado é diferente do código do email.

1. O sistema informa o Utilizador e termina o caso de uso.

Continuação da extensão cenário principal de sucesso:

5a. O Utilizador informa o sistema que pretende autenticar-se com o sensor de impressão digital.

1. Após o sistema operativo autenticar o Utilizador, o sistema compara o ID do dispositivo atual com o armazenado no sistema para verificar se está a ser feita uma autenticação duvidosa.
2. Se não existir nenhum ID do dispositivo armazenado, o sistema vai armazenar o atual, associando-o ao Utilizador.
3. O sistema autentica o Utilizador e continua no passo 8 (Cenário Principal de Sucesso).

Extensões (extensão 5a.):

1a. O sistema está a correr num sistema operativo que não suporta esta funcionalidade (<*Android* 6.0).

1. O sistema informa o Utilizador que não é possível aceder a este método de autenticação e continua no passo 6 (Cenário Principal de Sucesso).

1b. O sistema está a correr num dispositivo *hardware* que não possui sensor de impressão digital.

1. O sistema informa o Utilizador que não é possível aceder a este método de autenticação e continua no passo 6 (Cenário Principal de Sucesso).

1c. O sistema está a correr num dispositivo *hardware* que não possui nenhuma impressão digital registada no sistema operativo.

1. O sistema informa o Utilizador que não tem nenhuma impressão digital registada e continua o caso de uso no passo 6 (Cenário Principal de Sucesso).

1d. Os ID dos dispositivos de *hardware* são diferentes.

1. O sistema informa o Utilizador que existe uma autenticação duvidosa.
2. O sistema cria um código.
3. O sistema envia o código criado para o email do Utilizador.
4. O Utilizador digita o código do email e o sistema compara-o com o que foi enviado para o email do Utilizador.
5. O sistema armazena o novo ID do dispositivo, realiza a transferência e continua no passo 9 (Cenário Principal de Sucesso).

Extensões (extensão 5a.1d.):

4a. O código digitado é diferente do código do email.

1. O sistema informa o Utilizador e termina o caso de uso.

A.3 – UC3: Transferência interbancária

Ator Principal:

Utilizador.

Interesses:

Utilizador: Pretende efetuar uma transferência de uma das suas contas para outra conta interbancária.

Pré-Condições:

O Utilizador está autenticado perante o sistema.

Pós-Condições:

A transferência foi efetuada com sucesso.

Cenário Principal de Sucesso:

1. O Utilizador indica ao sistema que pretende efetuar uma transferência interbancária.
2. O Utilizador indica ao sistema que pretende ver as suas contas.
3. O sistema mostra as contas associadas ao Utilizador.
4. O Utilizador confirma a transferência preenchendo todos os campos necessários.
5. O Utilizador informa o sistema que pretende autenticar-se com o código pessoal.
6. O sistema mostra as quatro posições do código pessoal que o Utilizador tem de digitar.
7. O Utilizador indica os dígitos do código pessoal que estão nas posições pedidas pelo sistema.
8. O sistema compara os ID e realiza a transferência.
9. O sistema informa o Utilizador que a transferência foi realizada com sucesso.

Extensões (Cenário Principal de Sucesso):

4a. O Utilizador não preenche todos os campos necessários.

1. O sistema informa o Utilizador que deve preencher todos os campos necessários para poder realizar a transferência e o caso de uso termina.

4b. Os campos preenchidos pelo Utilizador estão inválidos.

1. O sistema informa o Utilizador que tem os campos errados, finaliza o caso de uso e pede-lhe que volte a introduzir os dados corretos para realizar a transferência.

8a. O sistema está indisponível para realizar a transferência.

1. O sistema deve informar o Utilizador que não é possível realizar a transferência atualmente e termina o caso de uso.

8b. Os ID dos dispositivos de *hardware* são diferentes.

1. O sistema informa o Utilizador que existe uma autenticação duvidosa.

2. O sistema cria um código.
3. O sistema envia o código criado para o email do Utilizador.
4. O Utilizador digita o código do email e o sistema compara-o com o que foi enviado para o email do Utilizador.
5. O sistema confirma a autenticação do Utilizador, realiza a transferência e continua no passo 9 (Cenário Principal de Sucesso).

Extensões (extensão 8b.):

4a. O código digitado é diferente do código do email.

1. O sistema informa o Utilizador e termina o caso de uso.

Continuação da extensão cenário principal de sucesso:

5a. O Utilizador informa o sistema que pretende autenticar-se com o sensor de impressão digital.

1. Após o sistema operativo autenticar o Utilizador, o sistema compara o ID do dispositivo atual com o armazenado no sistema para verificar se está a ser feita uma autenticação duvidosa.
2. Se não existir nenhum ID do dispositivo armazenado, o sistema vai armazenar o atual, associando-o ao Utilizador.
3. O sistema autentica o Utilizador e continua no passo 8 (Cenário Principal de Sucesso).

Extensões (extensão 5a.):

1a. O sistema está a correr num sistema operativo que não suporta esta funcionalidade (<*Android* 6.0).

1. O sistema informa o Utilizador que não é possível aceder a este método de autenticação e continua no passo 6 (Cenário Principal de Sucesso).

1b. O sistema está a correr num dispositivo *hardware* que não possui sensor de impressão digital.

1. O sistema informa o Utilizador que não é possível aceder a este método de autenticação e continua no passo 6 (Cenário Principal de Sucesso).

1c. O sistema está a correr num dispositivo *hardware* que não possui nenhuma impressão digital registada no sistema operativo.

1. O sistema informa o Utilizador que não tem nenhuma impressão digital registada e para proceder ao registo, continuando o caso de uso no passo 6 (Cenário Principal de Sucesso).

1d. Os ID dos dispositivos de *hardware* são diferentes.

1. O sistema informa o Utilizador que existe uma autenticação duvidosa.
2. O sistema cria um código.

3. O sistema envia o código criado para o email do Utilizador.
4. O Utilizador digita o código do email e o sistema compara-o com o que foi enviado para o email do Utilizador.
5. O sistema armazena o novo ID do dispositivo, realiza a transferência e continua no passo 8 (Cenário Principal de Sucesso).

Extensões (extensão 5a.1d.):

4a. O código digitado é diferente do código do email.

1. O sistema informa o Utilizador e termina o caso de uso.

A.4 – UC4: Transferência beneficiário entre contas do mesmo banco

Ator Principal:

Utilizador.

Interesses:

Utilizador: Pretende efetuar uma transferência de uma das suas contas para outra conta associada ao mesmo banco.

Pré-Condições:

O Utilizador está autenticado perante o sistema.

A transferência que o Utilizador vai realizar já deve estar registada no sistema.

Pós-Condições:

A transferência foi efetuada com sucesso.

Cenário Principal de Sucesso:

1. O Utilizador indica ao sistema que pretende efetuar uma transferência entre contas do mesmo banco.
2. O Utilizador indica ao sistema que pretende ver as suas transferências beneficiário.
3. O sistema mostra as transferências beneficiário associadas ao Utilizador.
4. O Utilizador indica que pretende efetuar uma determinada transferência.
5. O Utilizador informa o sistema que pretende autenticar-se com o código pessoal.
6. O sistema mostra as quatro posições do código pessoal que o Utilizador tem de digitar.
7. O Utilizador indica os dígitos do código pessoal que estão nas posições pedidas pelo sistema.
8. O sistema compara os ID e realiza a transferência.
9. O sistema informa o Utilizador que a transferência foi realizada com sucesso.

Extensões (Cenário Principal de Sucesso):

3a. O sistema não mostra as transferências, porque o Utilizador não tem transferências realizadas no sistema.

1. O sistema informa o Utilizador que não possui transferências realizadas e termina o caso de uso.

8a. O sistema não está disponível para a realização da transferência.

1. O sistema informa o Utilizador que não é possível realizar a transferência atualmente e finaliza o caso de uso.

8b. Os ID dos dispositivos de *hardware* são diferentes.

1. O sistema informa o Utilizador que existe uma autenticação duvidosa.

2. O sistema cria um código.
3. O sistema envia o código criado para o email do Utilizador.
4. O Utilizador digita o código do email e o sistema compara-o com o que foi enviado para o email do Utilizador.
5. O sistema confirma a autenticação do Utilizador, realiza a transferência e continua no passo 9 (Cenário Principal de Sucesso).

Extensões (extensão 8b.):

4a. O código digitado é diferente do código do email.

1. O sistema informa o Utilizador e termina o caso de uso.

Continuação da extensão cenário principal de sucesso:

5a. O Utilizador informa o sistema que pretende autenticar-se com o sensor de impressão digital.

1. Após o sistema operativo autenticar o Utilizador, o sistema compara o ID do dispositivo atual com o armazenado no sistema para verificar se está a ser feita uma autenticação duvidosa.
2. Se não existir nenhum ID do dispositivo armazenado, o sistema vai armazenar o atual, associando-o ao Utilizador.
3. O sistema autentica o Utilizador e continua no passo 8 (Cenário Principal de Sucesso).

Extensões (extensão 5a.):

1a. O sistema está a correr num sistema operativo que não suporta esta funcionalidade (<*Android* 6.0).

1. O sistema informa o Utilizador que não é possível aceder a este método de autenticação e continua no passo 6 (Cenário Principal de Sucesso).

1b. O sistema está a correr num dispositivo *hardware* que não possui sensor de impressão digital.

1. O sistema informa o Utilizador que não é possível aceder a este método de autenticação e continua no passo 6 (Cenário Principal de Sucesso).

1c. O sistema está a correr num dispositivo *hardware* que não possui nenhuma impressão digital registada no sistema operativo.

1. O sistema informa o Utilizador que não tem nenhuma impressão digital registada e continua no passo 6 (Cenário Principal de Sucesso).

1d. Os ID dos dispositivos de *hardware* são diferentes.

1. O sistema informa o Utilizador que existe uma autenticação duvidosa.
2. O sistema cria um código.
3. O sistema envia o código criado para o email do Utilizador.

4. O Utilizador digita o código do email e o sistema compara-o com o que foi enviado para o email do Utilizador.
5. O sistema armazena o novo ID do dispositivo, realiza a transferência e continua no passo 9 (Cenário Principal de Sucesso).

Extensões (extensão 5a.1d.):

4a. O código digitado é diferente do código do email.

1. O sistema informa o Utilizador e termina o caso de uso.

A.5 – UC5: Transferência frequente entre contas do mesmo banco

Ator Principal:

Utilizador.

Interesses:

Utilizador: Pretende efetuar uma transferência de uma das suas contas para outra conta associada ao mesmo banco.

Pré-Condições:

O Utilizador está autenticado perante o sistema.

A transferência que o Utilizador vai realizar já deve estar registada no sistema.

Pós-Condições:

A transferência foi efetuada com sucesso.

Cenário Principal de Sucesso:

1. O Utilizador indica ao sistema que pretende efetuar uma transferência entre contas do mesmo banco.
2. O Utilizador indica ao sistema que pretende ver as suas transferências frequentes.
3. O sistema mostra as transferências frequentes associadas ao Utilizador.
4. O Utilizador indica que pretende efetuar uma determinada transferência.
5. O Utilizador informa o sistema que pretende autenticar-se com o código pessoal.
6. O sistema mostra as quatro posições do código pessoal que o Utilizador tem de digitar.
7. O Utilizador indica os dígitos do código pessoal que estão nas posições pedidas pelo sistema.
8. O sistema compara os ID e realiza a transferência.
9. O sistema informa o Utilizador que a transferência foi realizada com sucesso.

Extensões (Cenário Principal de Sucesso):

3a. O sistema não mostra transferências, porque o Utilizador não tem transferências realizadas no sistema.

1. O sistema informa o Utilizador que não possui transferências realizadas e finaliza o caso de uso.

8a. O sistema não está disponível para a realização da transferência.

1. O sistema informa o Utilizador que não é possível realizar a transferência atualmente e termina o caso de uso.

8b. Os ID dos dispositivos de *hardware* são diferentes.

1. O sistema informa o Utilizador que existe uma autenticação duvidosa.

2. O sistema cria um código.
3. O sistema envia o código criado para o email do Utilizador.
4. O Utilizador digita o código do email e o sistema compara-o com o que foi enviado para o email do Utilizador.
5. O sistema confirma a autenticação do Utilizador, realiza a transferência e continua no passo 9 (Cenário Principal de Sucesso).

Extensões (extensão 8b.):

4a. O código digitado é diferente do código do email.

1. O sistema informa o Utilizador e termina o caso de uso.

Continuação da extensão cenário principal de sucesso:

5a. O Utilizador informa o sistema que pretende autenticar-se com o sensor de impressão digital.

1. Após o sistema operativo autenticar o Utilizador, o sistema compara o ID do dispositivo atual com o armazenado no sistema para verificar se está a ser feita uma autenticação duvidosa.
2. Se não existir nenhum ID do dispositivo armazenado, o sistema vai armazenar o atual, associando-o ao Utilizador.
3. O sistema autentica o Utilizador e continua no passo 8 (Cenário Principal de Sucesso).

Extensões (extensão 5a.):

1a. O sistema está a correr num sistema operativo que não suporta esta funcionalidade (<*Android* 6.0).

1. O sistema informa o Utilizador que não é possível aceder a este método de autenticação e continua no passo 6 (Cenário Principal de Sucesso).

1b. O sistema está a correr num dispositivo *hardware* que não possui sensor de impressão digital.

1. O sistema informa o Utilizador que não é possível aceder a este método de autenticação e continua no passo 6 (Cenário Principal de Sucesso).

1c. O sistema está a correr num dispositivo *hardware* que não possui nenhuma impressão digital registada no sistema operativo.

1. O sistema informa o Utilizador que não tem nenhuma impressão digital registada e continua no passo 6 (Cenário Principal de Sucesso).

1d. Os ID dos dispositivos de *hardware* são diferentes.

1. O sistema informa o Utilizador que existe uma autenticação duvidosa.
2. O sistema cria um código.
3. O sistema envia o código criado para o email do Utilizador.

4. O Utilizador digita o código do email e o sistema compara-o com o que foi enviado para o email do Utilizador.
5. O sistema armazena o novo ID do dispositivo, realiza a transferência e continua no passo 9 (Cenário Principal de Sucesso).

Extensões (5a.1d.):

4a. O código digitado é diferente do código do email.

1. O sistema informa o Utilizador e termina o caso de uso.

A.6 – UC6: Transferência beneficiário interbancária

Ator Principal:

Utilizador.

Interesses:

Utilizador: Pretende efetuar uma transferência de uma das suas contas para outra conta interbancária.

Pré-Condições:

O Utilizador está autenticado perante o sistema.

A transferência que o Utilizador vai realizar já deve estar registada no sistema.

Pós-Condições:

A transferência foi efetuada com sucesso.

Cenário Principal de Sucesso:

1. O Utilizador indica ao sistema que pretende efetuar uma transferência interbancária.
2. O Utilizador indica ao sistema que pretende ver as suas transferências beneficiário.
3. O sistema mostra as transferências beneficiário associadas ao Utilizador.
4. O Utilizador indica que pretende efetuar uma determinada transferência.
5. O Utilizador informa o sistema que pretende autenticar-se com o código pessoal.
6. O sistema mostra as quatro posições do código pessoal que o Utilizador tem de digitar.
7. O Utilizador indica os dígitos do código pessoal que estão nas posições pedidas pelo sistema.
8. O sistema compara os ID e realiza a transferência.
9. O sistema informa o Utilizador que a transferência foi realizada com sucesso.

Extensões (Cenário Principal de Sucesso):

3a. O sistema não mostra transferências, porque o Utilizador não tem transferências realizadas no sistema.

1. O sistema informa o Utilizador que não possui transferências realizadas e finaliza o caso de uso.

8a. O sistema não está disponível para a realização da transferência.

1. O sistema informa o Utilizador que não é possível realizar a transferência atualmente e termina o caso de uso.

8b. Os ID dos dispositivos de *hardware* são diferentes.

1. O sistema informa o Utilizador que existe uma autenticação duvidosa.
2. O sistema cria um código.

3. O sistema envia o código criado para o email do Utilizador.
4. O Utilizador digita o código do email e o sistema compara-o com o que foi enviado para o email do Utilizador.
5. O sistema confirma a autenticação do Utilizador, realiza a transferência e continua no passo 9 (Cenário Principal de Sucesso).

Extensões (extensão 8b.):

4a. O código digitado é diferente do código do email.

1. O sistema informa o Utilizador e termina o caso de uso.

Continuação da extensão cenário principal de sucesso:

5a. O Utilizador informa o sistema que pretende autenticar-se com o sensor de impressão digital.

1. Após o sistema operativo autenticar o Utilizador, o sistema compara o ID do dispositivo atual com o armazenado no sistema para verificar se está a ser feita uma autenticação duvidosa.
2. Se não existir nenhum ID do dispositivo armazenado, o sistema vai armazenar o atual, associando-o ao Utilizador.
3. O sistema autentica o Utilizador e continua no passo 8 (Cenário Principal de Sucesso).

Extensões (extensão 5a.):

1a. O sistema está a correr num sistema operativo que não suporta esta funcionalidade (<*Android* 6.0).

1. O sistema informa o Utilizador que não é possível aceder a este método de autenticação e continua no passo 6 (Cenário Principal de Sucesso).

1b. O sistema está a correr num dispositivo *hardware* que não possui sensor de impressão digital.

1. O sistema informa o Utilizador que não é possível aceder a este método de autenticação e continua no passo 6 (Cenário Principal de Sucesso).

1c. O sistema está a correr num dispositivo *hardware* que não possui nenhuma impressão digital registada no sistema operativo.

1. O sistema informa o Utilizador que não tem nenhuma impressão digital registada e continua no passo 6 (Cenário Principal de Sucesso).

1d. Os ID dos dispositivos de *hardware* são diferentes.

1. O sistema informa o Utilizador que existe uma autenticação duvidosa.
2. O sistema cria um código.
3. O sistema envia o código criado para o email do Utilizador.

4. O Utilizador digita o código do email e o sistema compara-o com o que foi enviado para o email do Utilizador.
5. O sistema armazena o novo ID do dispositivo, realiza a transferência e continua no passo 9 (Cenário Principal de Sucesso).

Extensões (extensão 5a.1d.):

4a. O código digitado é diferente do código do email.

1. O sistema informa o Utilizador e termina o caso de uso.

A.7 – UC7: Transferência frequente interbancária

Ator Principal:

Utilizador.

Interesses:

Utilizador: Pretende efetuar uma transferência de uma das suas contas para outra conta interbancária.

Pré-Condições:

O Utilizador está autenticado perante o sistema.

A transferência que o Utilizador vai realizar já deve estar registada no sistema.

Pós-Condições:

A transferência foi efetuada com sucesso.

Cenário Principal de Sucesso:

1. O Utilizador indica ao sistema que pretende efetuar uma transferência interbancária.
2. O Utilizador indica ao sistema que pretende ver as suas transferências frequentes.
3. O sistema mostra as transferências frequentes associadas ao Utilizador.
4. O Utilizador indica que pretende efetuar uma determinada transferência.
5. O Utilizador informa o sistema que pretende autenticar-se com o código pessoal.
6. O sistema mostra as quatro posições do código pessoal que o Utilizador tem de digitar.
7. O Utilizador indica os dígitos do código pessoal que estão nas posições pedidas pelo sistema.
8. O sistema compara os ID e realiza a transferência.
9. O sistema informa o Utilizador que a transferência foi realizada com sucesso.

Extensões (Cenário Principal de Sucesso):

3a. O sistema não mostra transferências, porque o Utilizador não tem transferências realizadas no sistema.

1. O sistema informa o Utilizador que não possui transferências realizadas e termina o caso de uso.

8a. O sistema não está disponível para a realização da transferência.

1. O sistema informa o Utilizador que não é possível realizar a transferência atualmente e finaliza o caso de uso.

8b. Os ID dos dispositivos de *hardware* são diferentes.

1. O sistema informa o Utilizador que existe uma autenticação duvidosa.
2. O sistema cria um código.

3. O sistema envia o código criado para o email do Utilizador.
4. O Utilizador digita o código do email e o sistema compara-o com o que foi enviado para o email do Utilizador.
5. O sistema confirma a autenticação do Utilizador, realiza a transferência e continua no passo 9 (Cenário Principal de Sucesso).

Extensões (extensão 8b.):

4a. O código digitado é diferente do código do email.

1. O sistema informa o Utilizador e termina o caso de uso.

Continuação da extensão cenário principal de sucesso:

5a. O Utilizador informa o sistema que pretende autenticar-se com o sensor de impressão digital.

1. Após o sistema operativo autenticar o Utilizador, o sistema compara o ID do dispositivo atual com o armazenado no sistema para verificar se está a ser feita uma autenticação duvidosa.
2. Se não existir nenhum ID do dispositivo armazenado, o sistema vai armazenar o atual, associando-o ao Utilizador.
3. O sistema autentica o Utilizador e continua no passo 8 (Cenário Principal de Sucesso).

Extensões (extensão 5a.):

1a. O sistema está a correr num sistema operativo que não suporta esta funcionalidade (<*Android* 6.0).

1. O sistema informa o Utilizador que não é possível aceder a este método de autenticação e continua no passo 6 (Cenário Principal de Sucesso).

1b. O sistema está a correr num dispositivo *hardware* que não possui sensor de impressão digital.

1. O sistema informa o Utilizador que não é possível aceder a este método de autenticação e continua no passo 6 (Cenário Principal de Sucesso).

1c. O sistema está a correr num dispositivo *hardware* que não possui nenhuma impressão digital registada no sistema operativo.

1. O sistema informa o Utilizador que não tem nenhuma impressão digital registada e continua no passo 6 (Cenário Principal de Sucesso).

1d. Os ID dos dispositivos de *hardware* são diferentes.

1. O sistema informa o Utilizador que existe uma autenticação duvidosa.
2. O sistema cria um código.
3. O sistema envia o código criado para o email do Utilizador.

4. O Utilizador digita o código do email e o sistema compara-o com o que foi enviado para o email do Utilizador.
5. O sistema armazena o novo ID do dispositivo, realiza a transferência e continua no passo 9 (Cenário Principal de Sucesso).

Extensões (extensão 5a.1d.):

4a. O código digitado é diferente do código do email.

1. O sistema informa o Utilizador e termina o caso de uso.

A.8 – UC8: Cria Transferência por código

Ator Principal:

Utilizador.

Interesses:

Utilizador: Pretende criar uma transferência de uma das suas contas para outra conta do mesmo banco através de um código.

Pré-Condições:

O Utilizador está autenticado perante o sistema.

Pós-Condições:

O código foi criado com sucesso.

Cenário Principal de Sucesso:

1. O Utilizador indica ao sistema que pretende efetuar uma transferência por código.
2. O Utilizador indica ao sistema a conta de origem, o montante e a descrição da Transferência.
3. O Utilizador informa o sistema que pretende autenticar-se com o código pessoal.
4. O sistema mostra as quatro posições do código pessoal que o Utilizador tem de digitar.
5. O Utilizador indica os dígitos do código pessoal que estão nas posições pedidas pelo sistema.
6. O sistema compara os ID e armazena a Transferência criada.
7. O sistema informa o utilizador que o código com os dados da Transferência foi criado com sucesso.

Extensões (Cenário Principal de Sucesso):

2a. O montante da Transferência é superior ao saldo da conta.

1. O sistema informa que o montante é superior ao saldo da conta.

3a. O Utilizador informa o sistema que pretende autenticar-se com o sensor de impressão digital.

1. Após o sistema operativo autenticar o Utilizador, o sistema compara o ID do dispositivo atual com o armazenado no sistema para verificar se está a ser feita uma autenticação duvidosa.
2. Se não existir nenhum ID do dispositivo armazenado, o sistema vai armazenar o atual, associando-o ao Utilizador.
3. O sistema autentica o Utilizador e continua no passo 6 (Cenário Principal de Sucesso).

Extensões (extensão 3a.):

1a. O sistema está a correr num sistema operativo que não suporta esta funcionalidade (<*Android* 6.0).

1. O sistema informa o Utilizador que não é possível aceder a este método de autenticação e continua no passo 6 (Cenário Principal de Sucesso).

1b. O sistema está a correr num dispositivo *hardware* que não possui sensor de impressão digital.

1. O sistema informa o Utilizador que não é possível aceder a este método de autenticação e continua no passo 6 (Cenário Principal de Sucesso).

1c. O sistema está a correr num dispositivo *hardware* que não possui nenhuma impressão digital registada no sistema operativo.

1. O sistema informa o Utilizador que não tem nenhuma impressão digital registada e continua no passo 6 (Cenário Principal de Sucesso).

1d. Os ID dos dispositivos de *hardware* são diferentes.

1. O sistema informa o Utilizador que existe uma autenticação duvidosa.
2. O sistema cria um código.
3. O sistema envia o código criado para o email do Utilizador.
4. O Utilizador digita o código do email e o sistema compara-o com o que foi enviado para o email do Utilizador.
5. O sistema armazena o novo ID do dispositivo e continua no passo 8 (Cenário Principal de Sucesso).

Extensões (extensão 3a.1d.):

4a. O código digitado é diferente do código do email.

1. O sistema informa o Utilizador e termina o caso de uso.

Continuação da extensão cenário principal de sucesso:

6a. Os ID dos dispositivos de *hardware* são diferentes.

1. O sistema informa o Utilizador que existe uma autenticação duvidosa.
2. O sistema cria um código.
3. O sistema envia o código criado para o email do Utilizador.
4. O Utilizador digita o código do email e o sistema compara-o com o que foi enviado para o email do Utilizador.
5. O sistema confirma a autenticação do Utilizador, armazena a Transferência criada e continua no passo 7 (Cenário Principal de Sucesso).

Extensões (extensão 6a.):

4a. O código digitado é diferente do código do email.

1. O sistema informa o Utilizador e termina o caso de uso.

A.9 – UC9: Efetua Transferência por código

Ator Principal:

Utilizador.

Interesses:

Utilizador: Pretende efetuar uma transferência de uma das suas contas para outra conta do mesmo banco através de um código.

Pré-Condições:

O Utilizador está autenticado perante o sistema.

Existe um código criado.

Pós-Condições:

A Transferência foi realizada com sucesso.

Cenário Principal de Sucesso:

1. O Utilizador indica ao sistema que pretende efetuar uma transferência com código.
2. O Utilizador indica ao sistema que pretende ler o código (código associado àquela Transferência).
3. O sistema vai procurar os dados da Transferência.
4. O sistema informa o Utilizador dos dados da Transferência.
5. O Utilizador indica que aceita a Transferência e introduz a conta de destino.
6. O sistema informa o Utilizador que a Transferência foi realizada com sucesso.

Extensões (Cenário Principal de Sucesso):

2a. O código que o Utilizador escolheu já não se encontra disponível.

1. O sistema informa o Utilizador que o código está indisponível para a Transferência e termina o caso de uso.

3a. O sistema não está disponível para realizar a Transferência.

1. O sistema informa que não foi possível realizar a Transferência e termina o caso de uso.

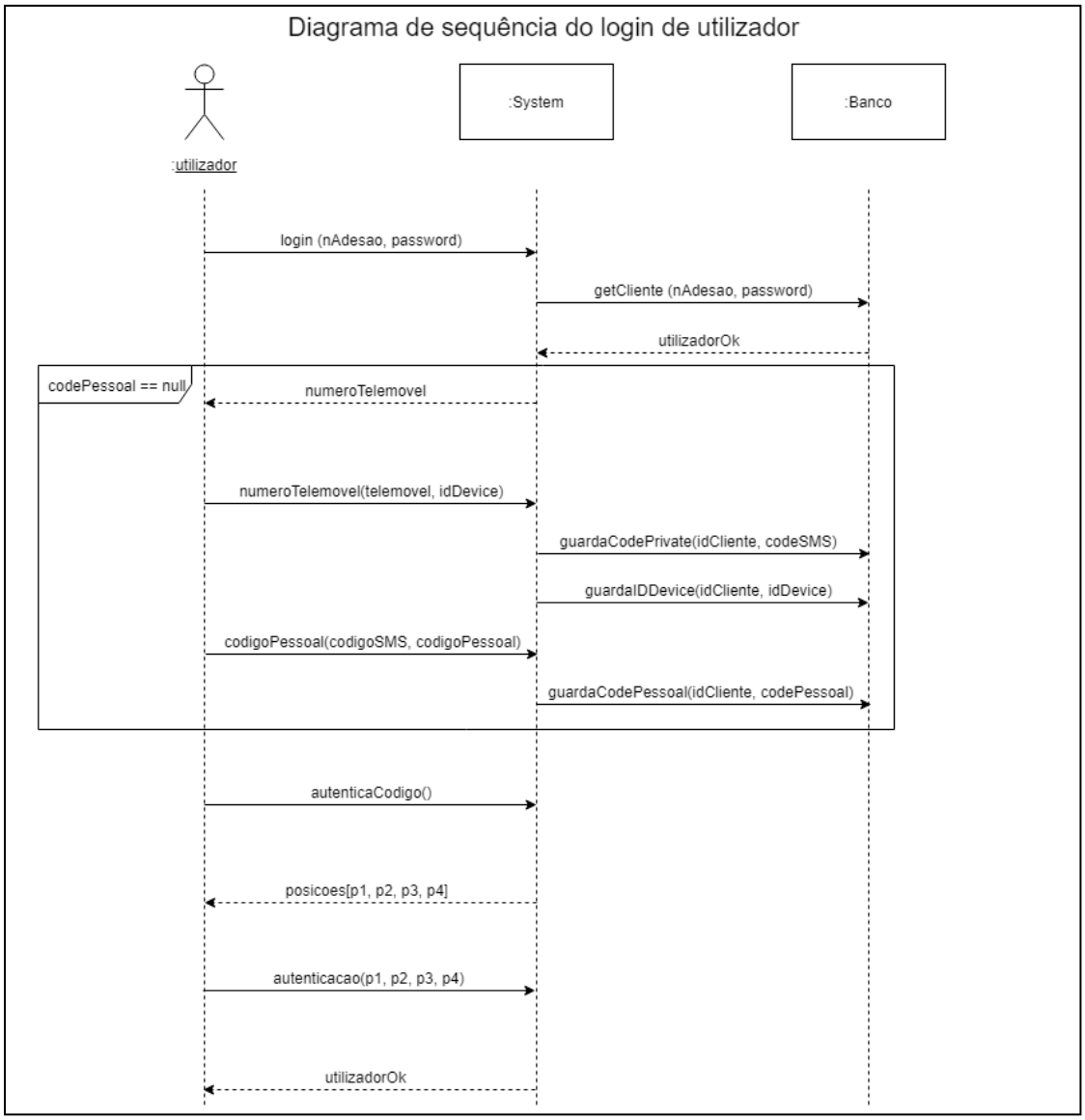
4a. O Utilizador indica que rejeita a Transferência.

1. O sistema informa que a Transferência foi removida do sistema e termina o caso de uso.

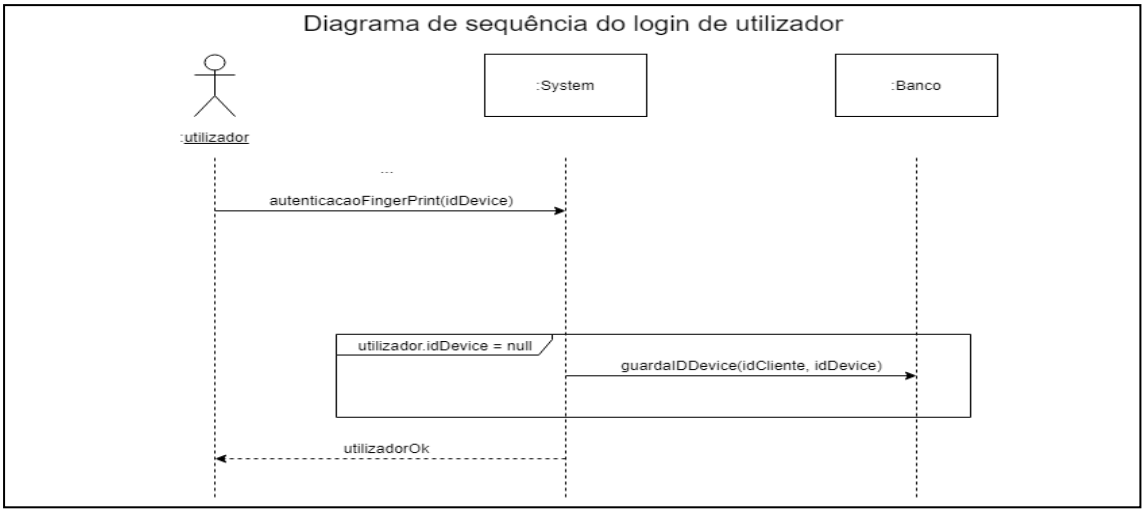
Anexo B – Diagramas de sequência

B.1 – UC1: Login de utilizador

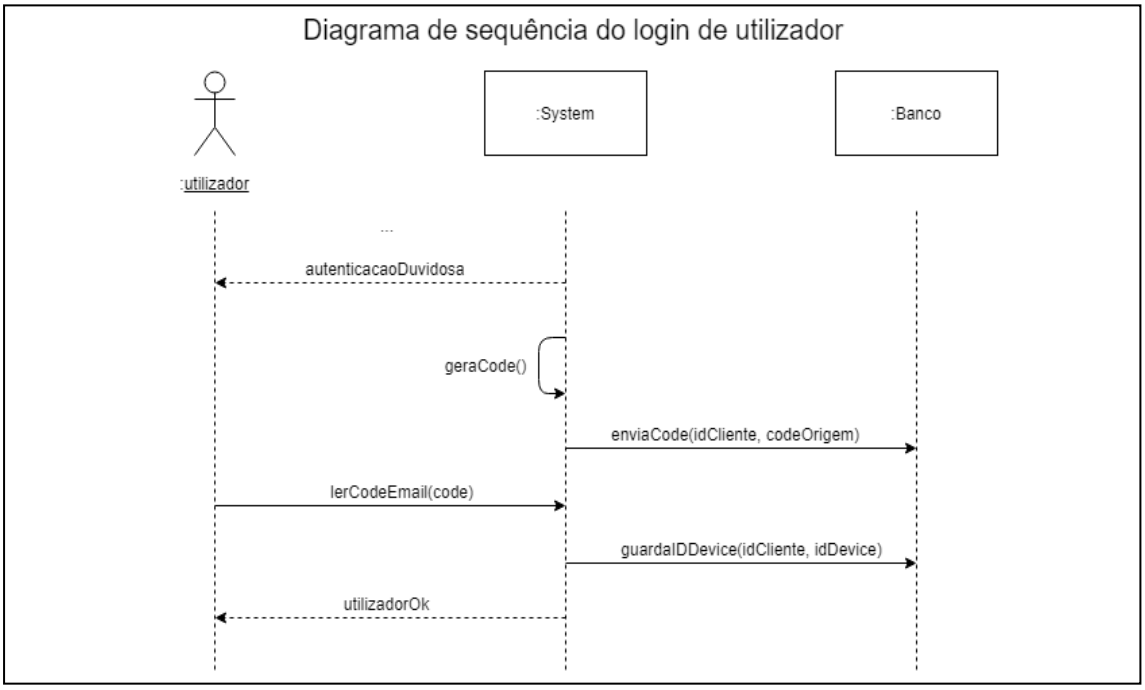
Login de utilizador utilizando as quatro posições do código pessoal como processo de autenticação.



Extensão do login de utilizador utilizando a impressão digital como processo de autenticação.



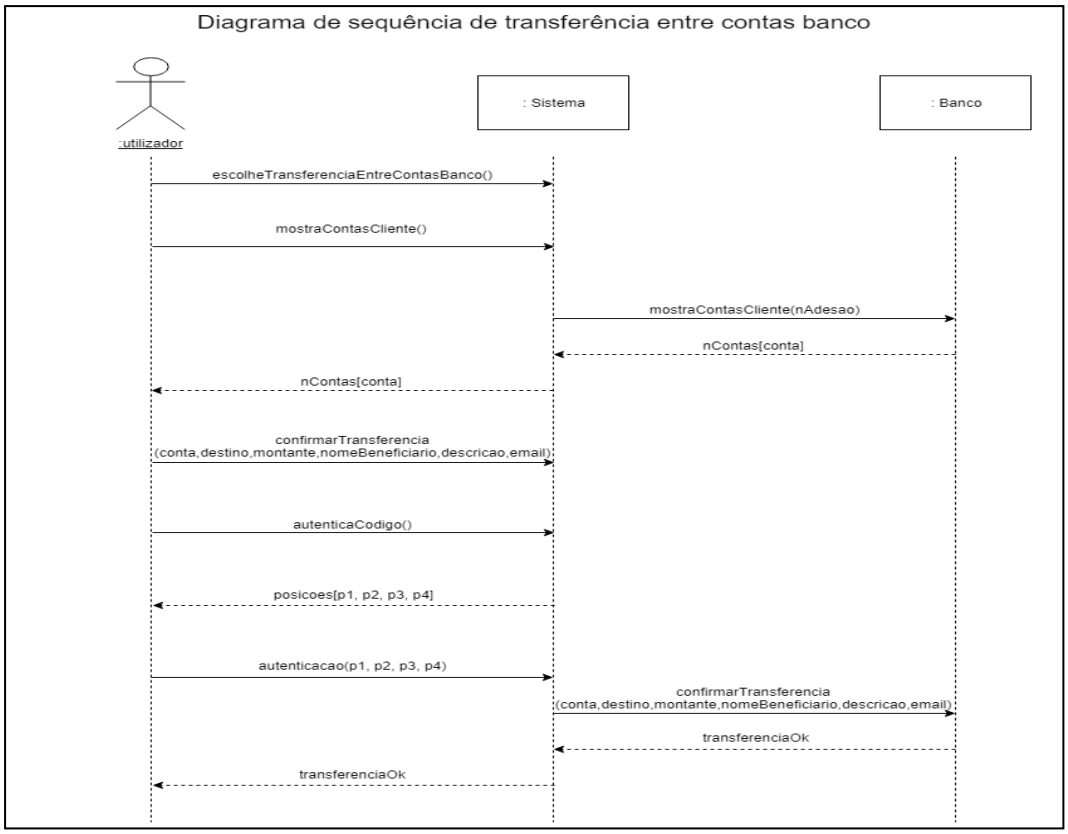
Extensão do login de utilizador quando o sistema deteta uma autenticação duvidosa.



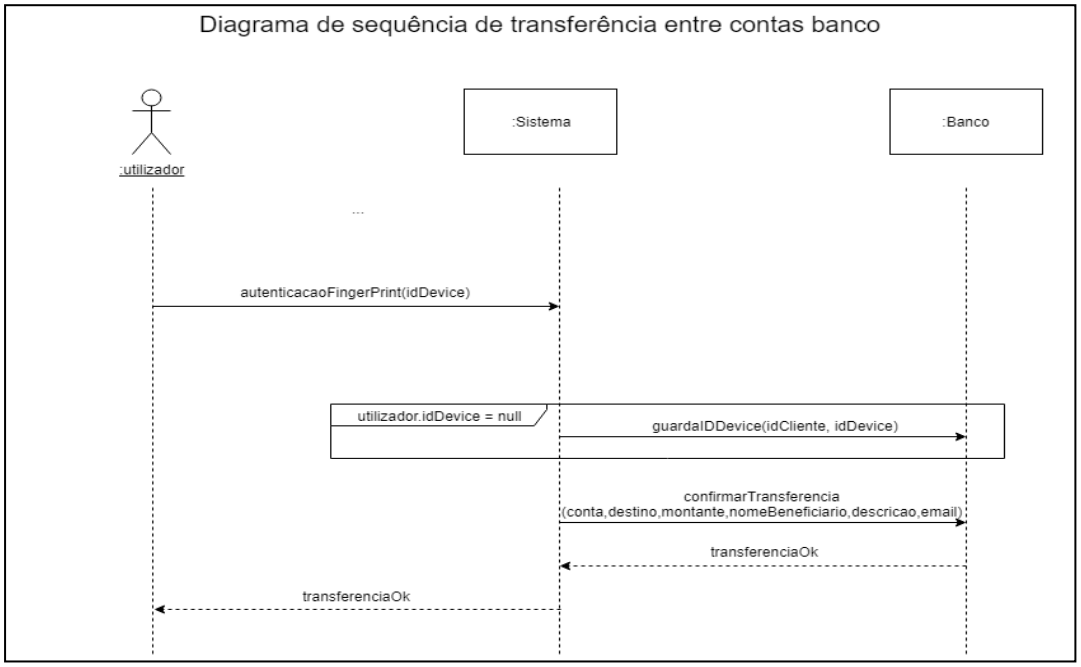
(https://gitlab.com/Henry15/Imagens_da_tese/tree/master/Diagramas_de_sequencia/Login)

B.2 – UC2: Transferência entre contas do mesmo banco

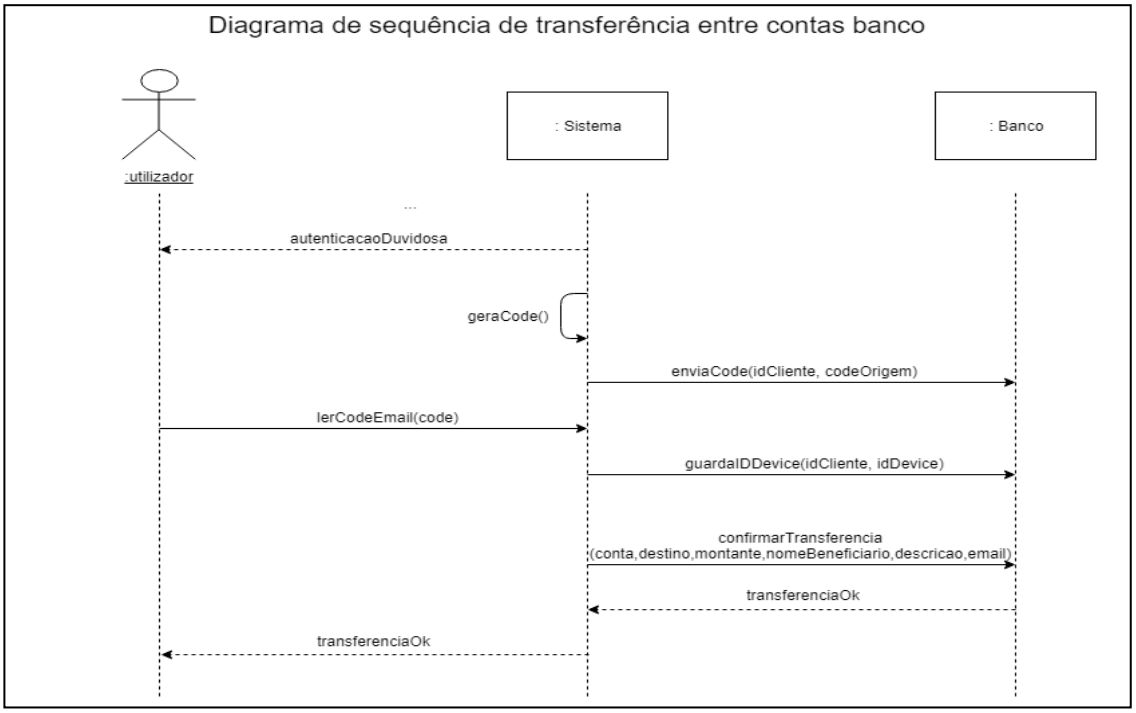
Realização de uma transferência entre contas do mesmo banco utilizando as quatro posições do código pessoal como processo de autenticação.



Extensão da realização de uma transferência entre contas do mesmo banco utilizando a impressão digital como processo de autenticação.



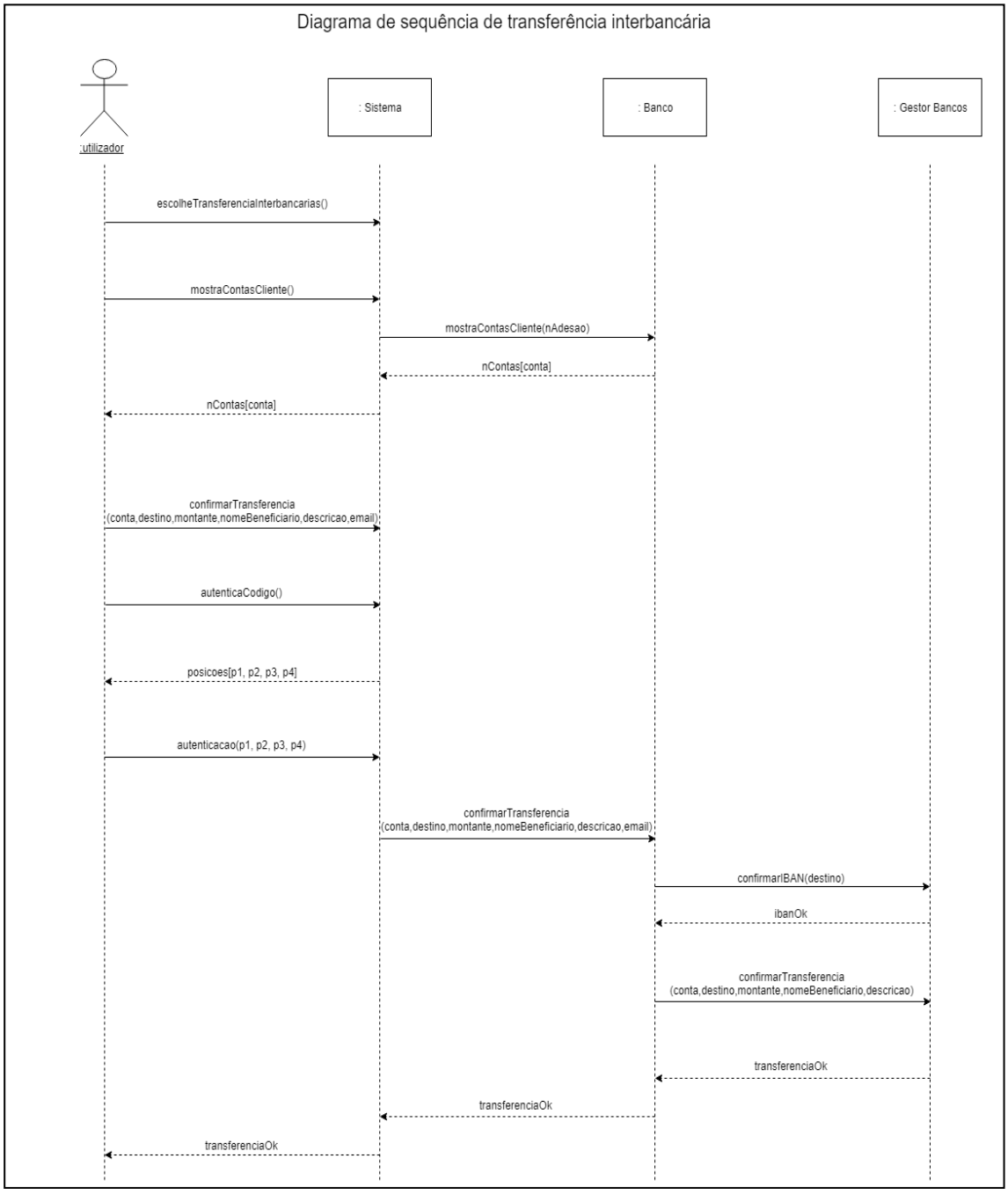
Extensão da realização de uma transferência entre contas do mesmo banco quando o sistema deteta uma autenticação duvidosa.



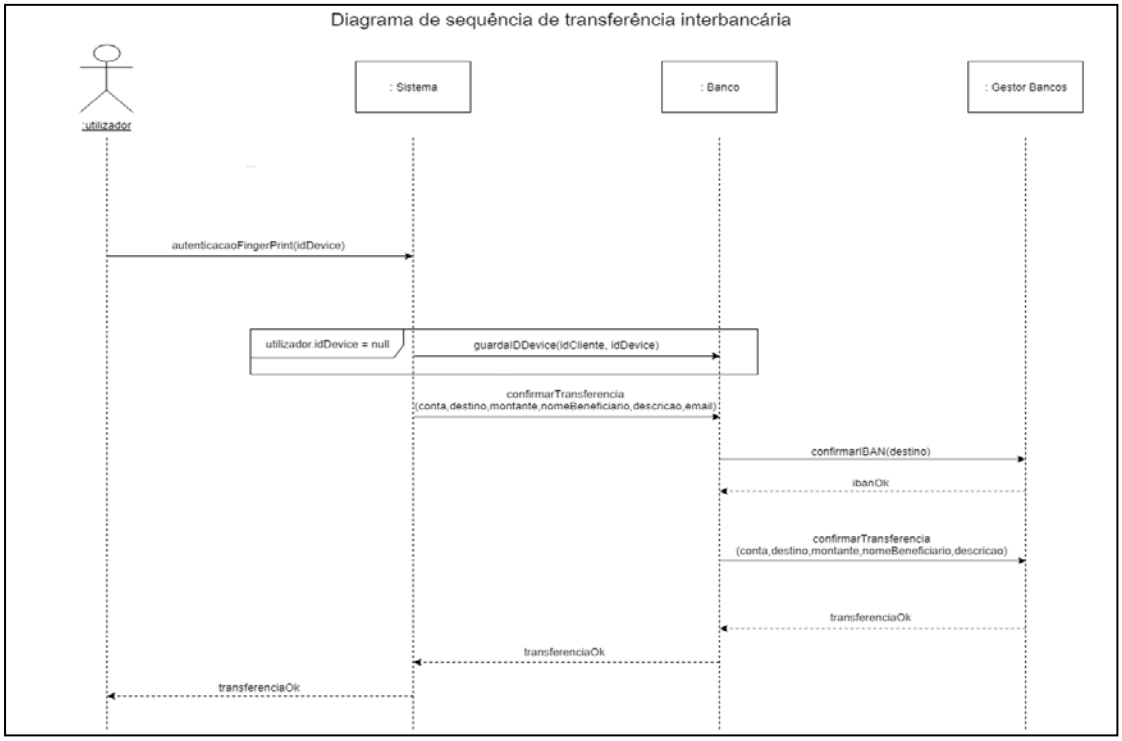
(https://gitlab.com/Henry15/Imagens_da_tese/tree/master/Diagramas_de_sequencia/TransferenciaECB)

B.3 – UC3: Transferência interbancária

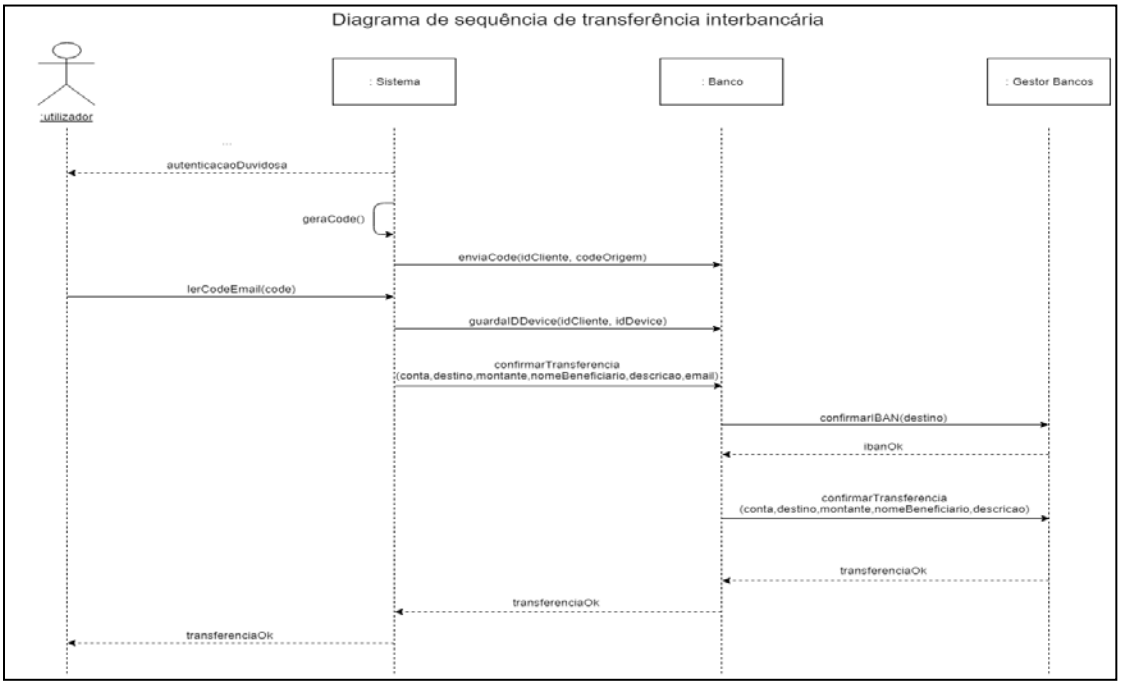
Realização de uma transferência interbancária utilizando as quatro posições do código pessoal como processo de autenticação.



Extensão da realização de uma transferência interbancária utilizando a impressão digital como processo de autenticação.



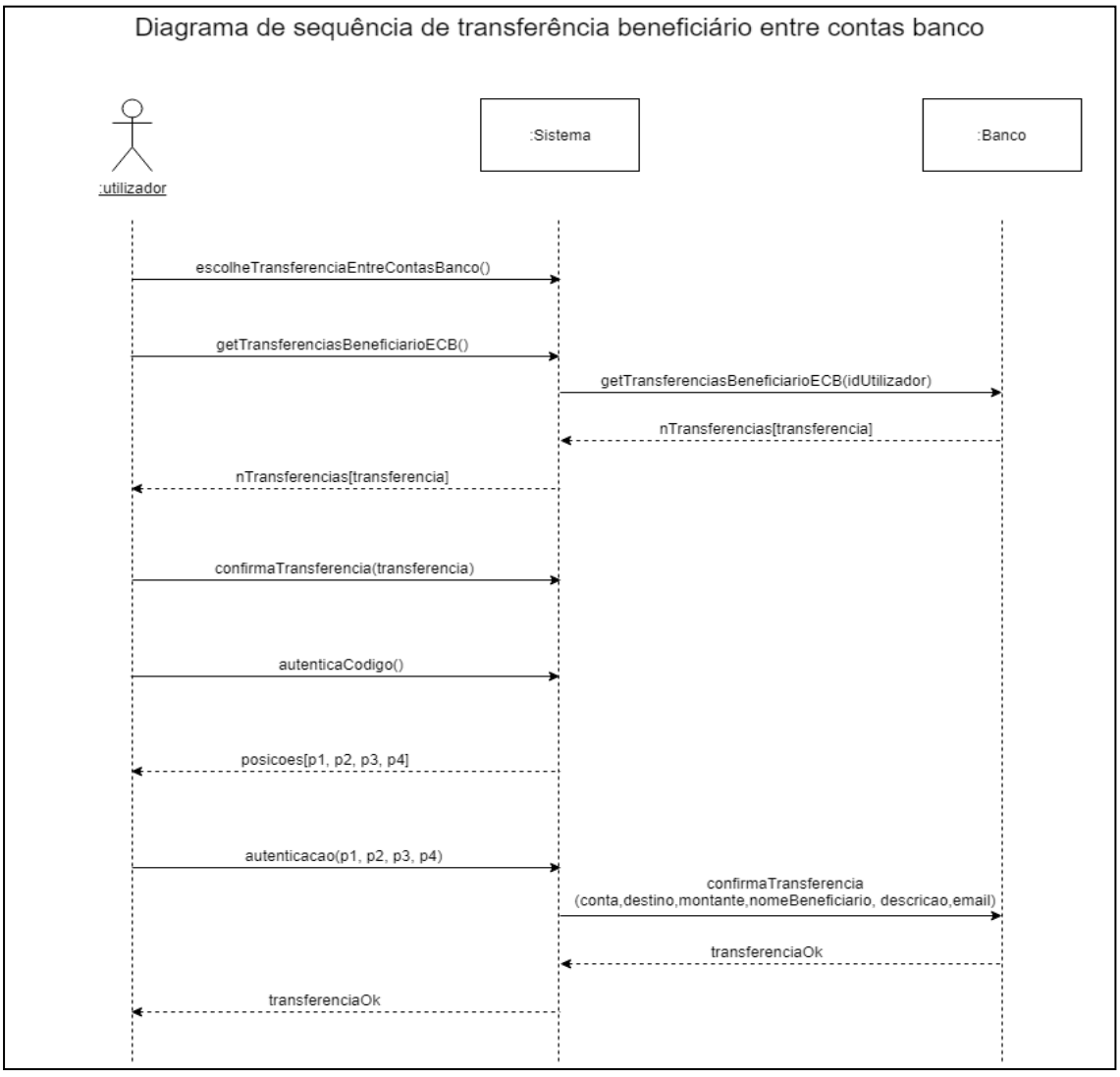
Extensão da realização de uma transferência interbancária quando o sistema deteta uma autenticação duvidosa.



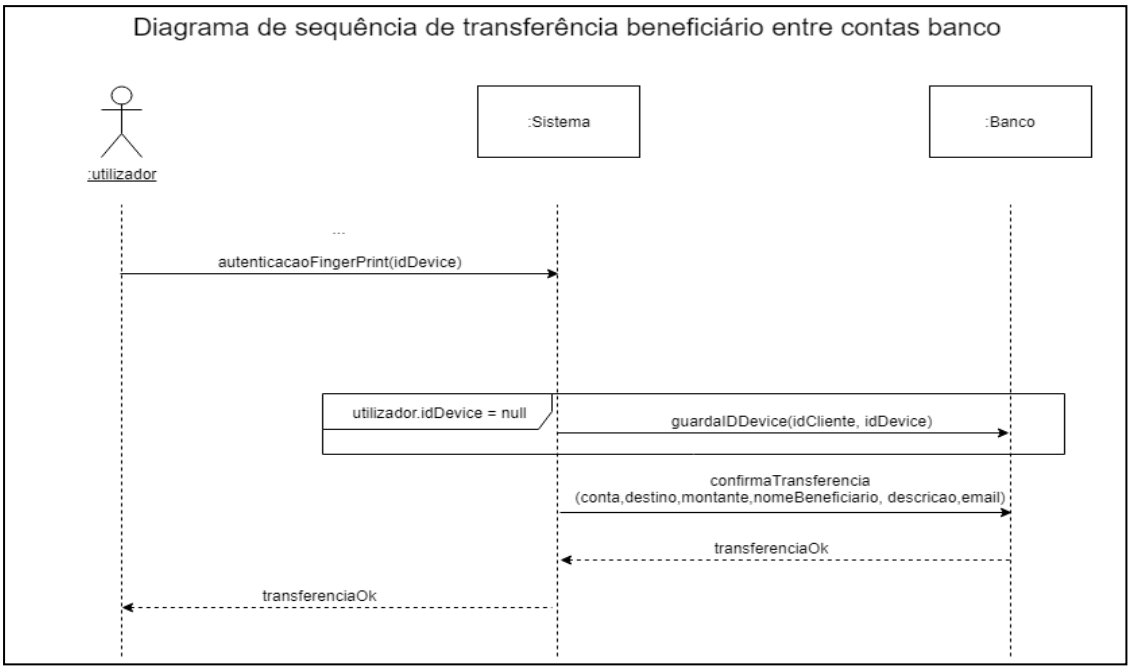
(https://gitlab.com/Henry15/Imagens_da_tese/tree/master/Diagramas_de_sequencia/TransferenciaInter)

B.4 – UC4: Transferência beneficiário entre contas do mesmo banco

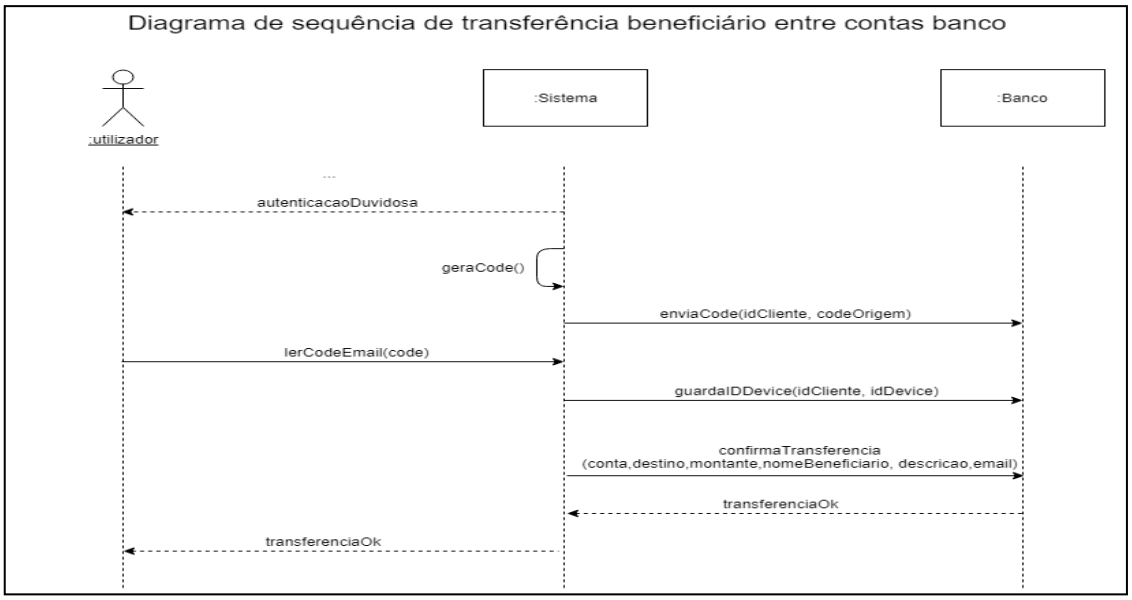
Realização de uma transferência entre contas do mesmo banco já efetuada anteriormente, ordenada por ordem do beneficiário, utilizando as quatro posições do código pessoal como processo de autenticação.



Extensão da realização de uma transferência entre contas do mesmo banco já efetuada anteriormente, ordenada por ordem do beneficiário, utilizando a impressão digital como processo de autenticação.



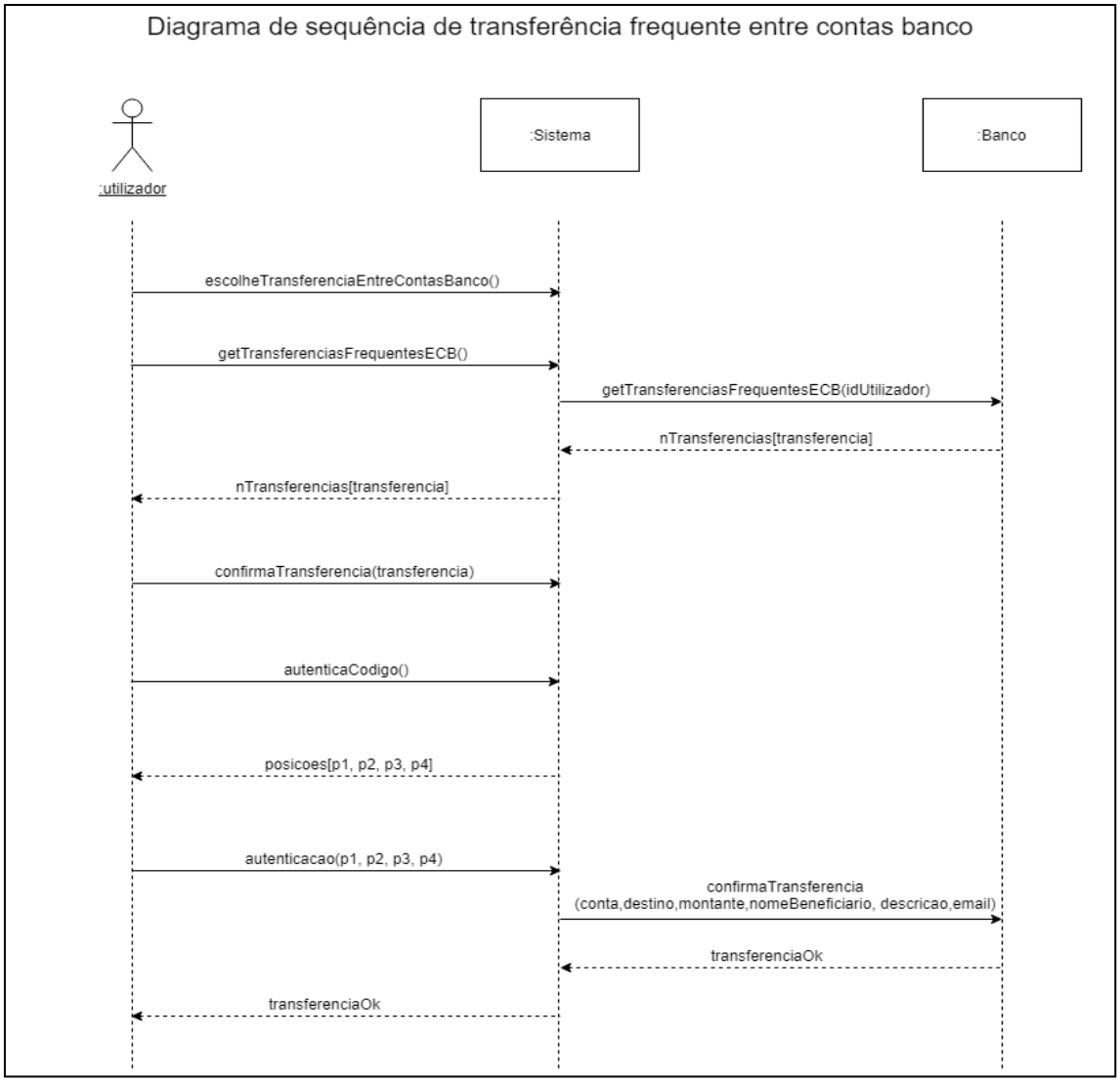
Extensão da realização de uma transferência entre contas do mesmo banco já efetuada anteriormente, ordenada por ordem do beneficiário, quando o sistema deteta uma autenticação duvidosa.



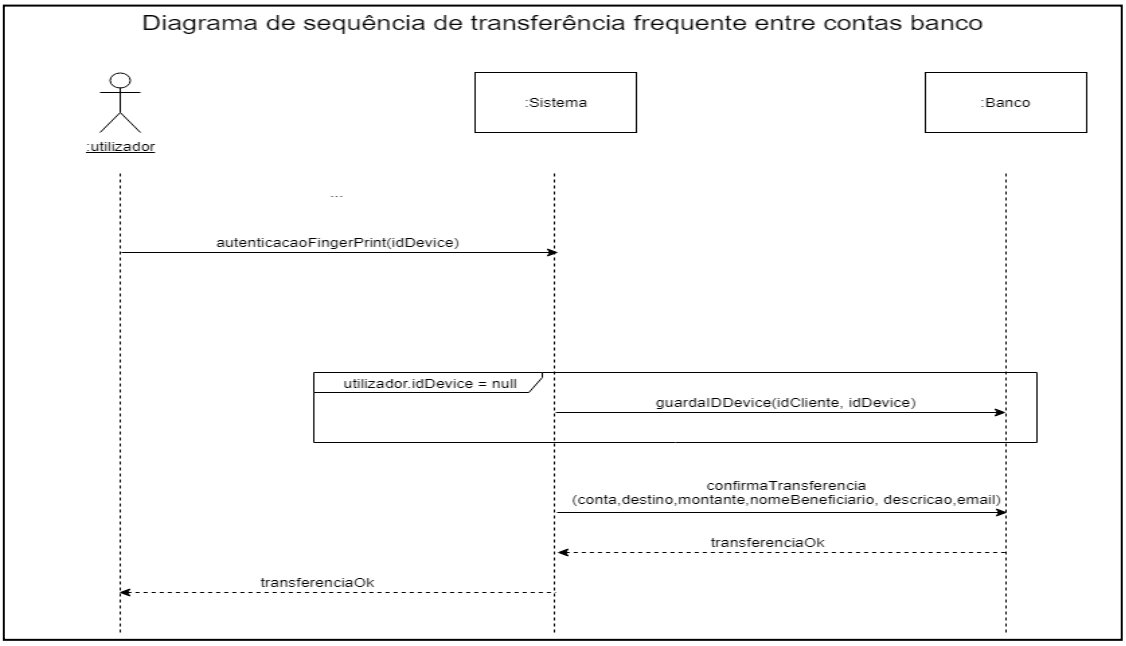
(https://gitlab.com/Henry15/Imagens_da_tese/tree/master/Diagramas_de_sequencia/TransferenciaECB/Beneficiario)

B.5 – UC5: Transferência frequente entre contas do mesmo banco

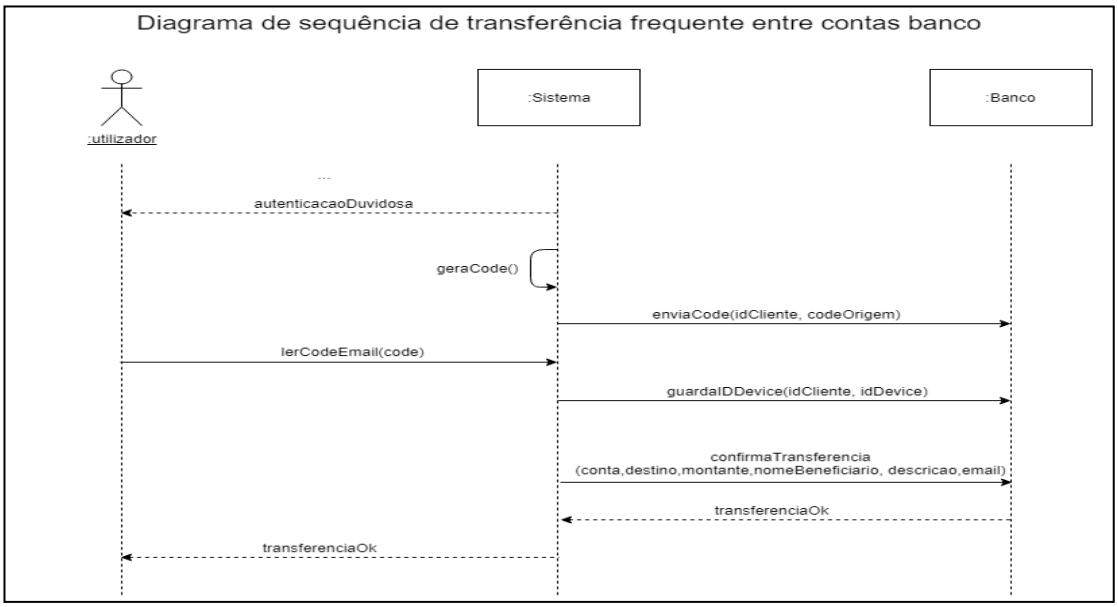
Realização de uma transferência entre contas do mesmo banco já efetuada anteriormente, ordenada por ordem de frequência, utilizando as quatro posições do código pessoal como processo de autenticação.



Extensão da realização de uma transferência entre contas do mesmo banco já efetuada anteriormente, ordenada por ordem de frequência, utilizando a impressão digital como processo de autenticação.



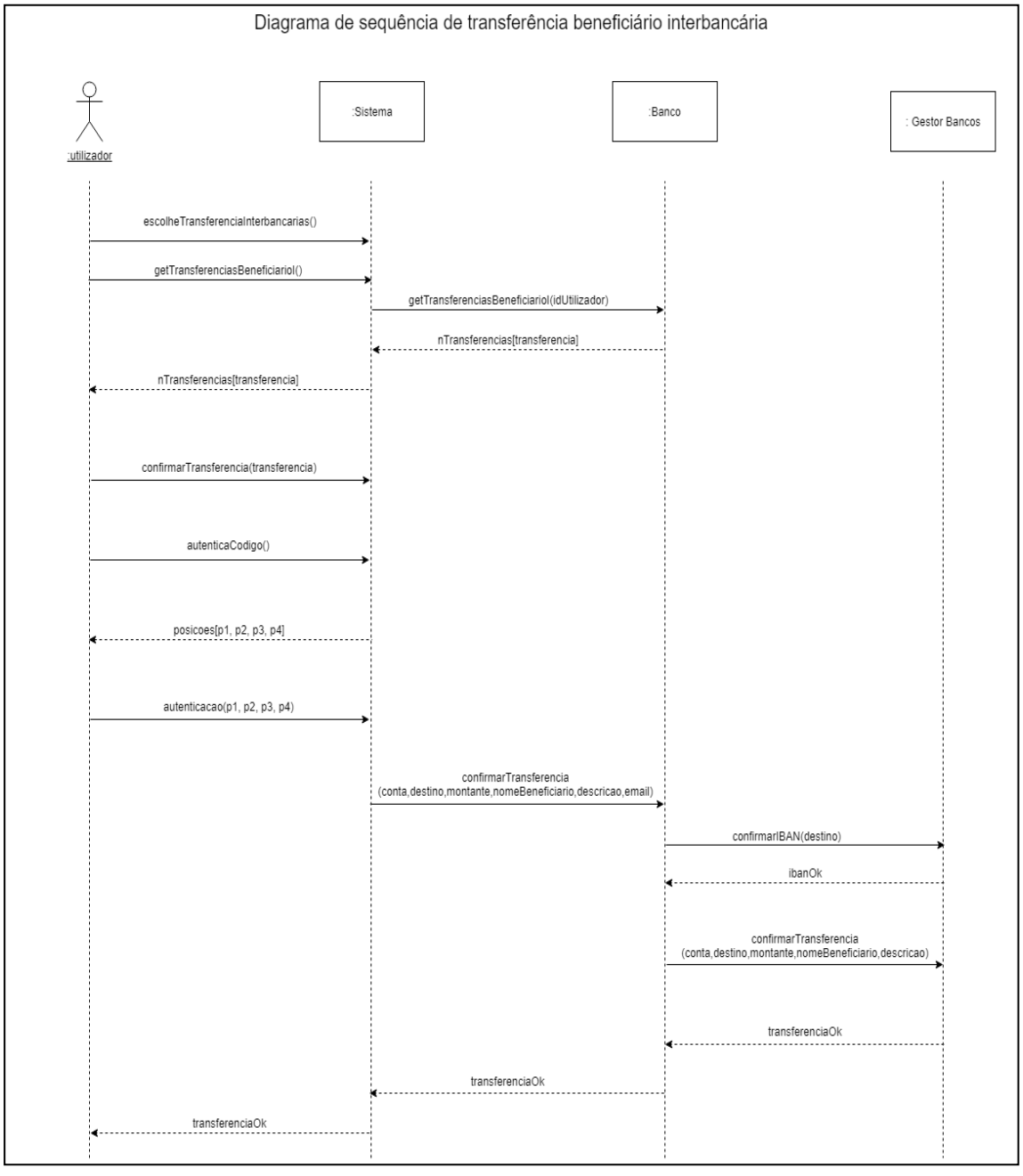
Extensão da realização de uma transferência entre contas do mesmo banco já efetuada anteriormente, ordenada por ordem de frequência, quando o sistema deteta uma autenticação duvidosa.



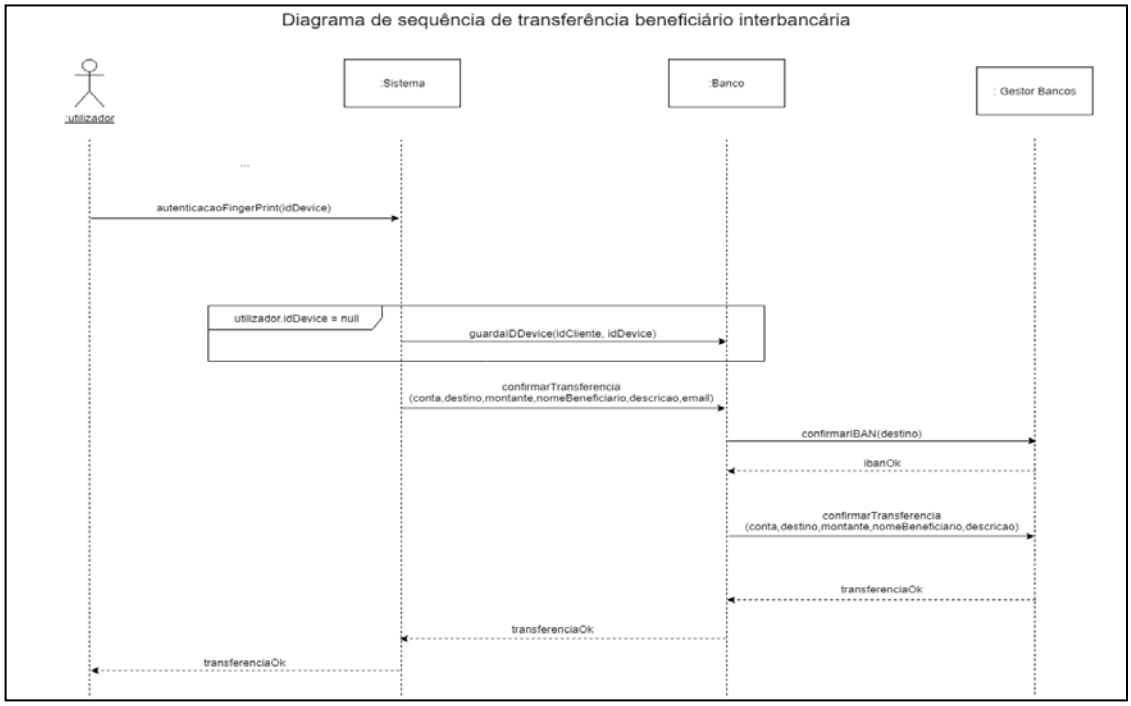
(https://gitlab.com/Henry15/Imagens_da_tese/tree/master/Diagramas_de_sequencia/TransferenciaECB/Frequente)

B.6 – UC6: Transferência beneficiário interbancária

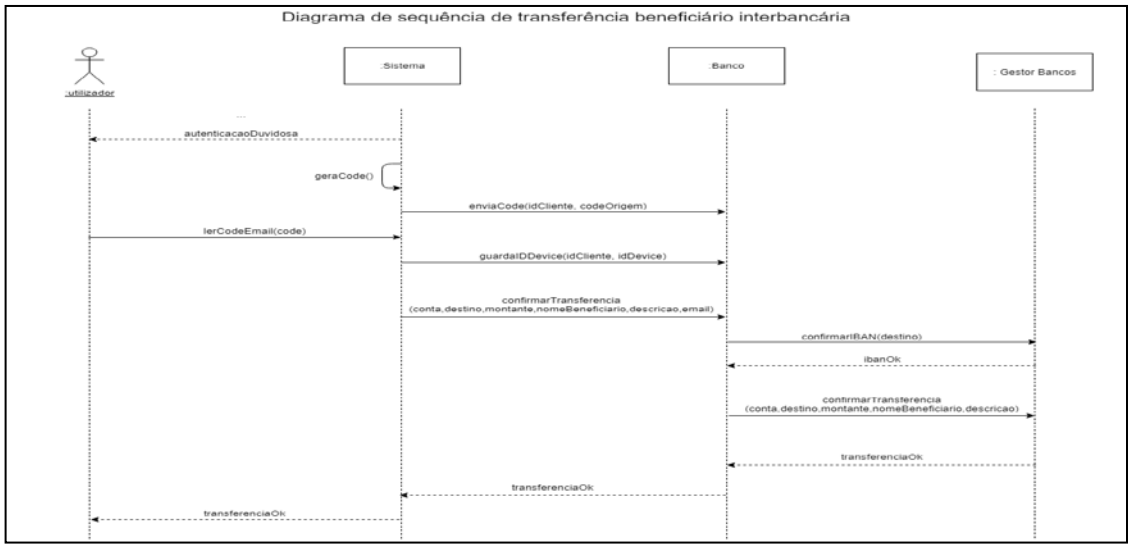
Realização de uma transferência interbancária já efetuada anteriormente, ordenada por ordem do beneficiário, utilizando as quatro posições do código pessoal como processo de autenticação.



Extensão da realização de uma transferência interbancária já efetuada anteriormente, ordenada por ordem do beneficiário, utilizando a impressão digital como processo de autenticação.



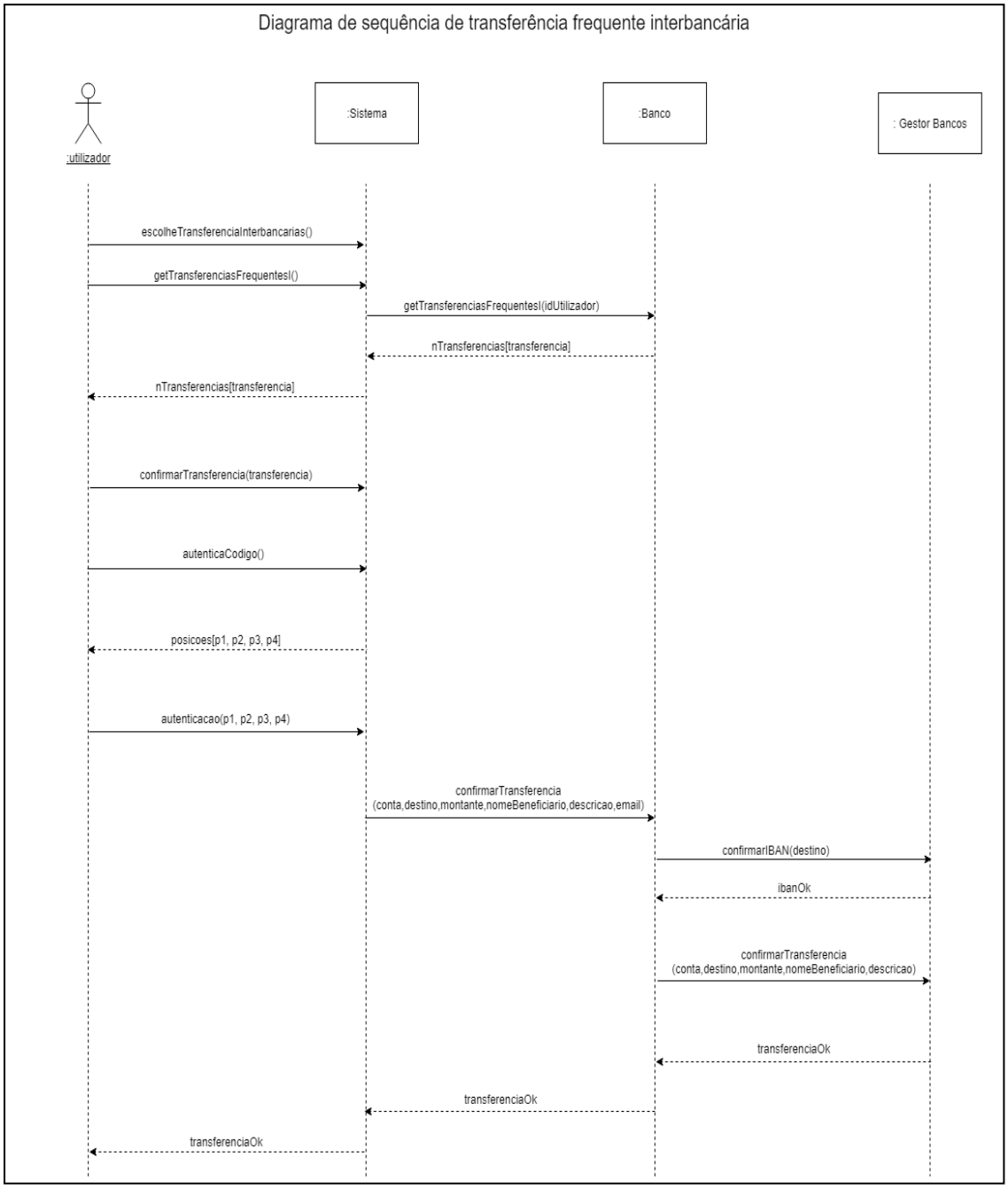
Extensão da realização de uma transferência interbancária já efetuada anteriormente, ordenada por ordem do beneficiário, quando o sistema deteta uma autenticação



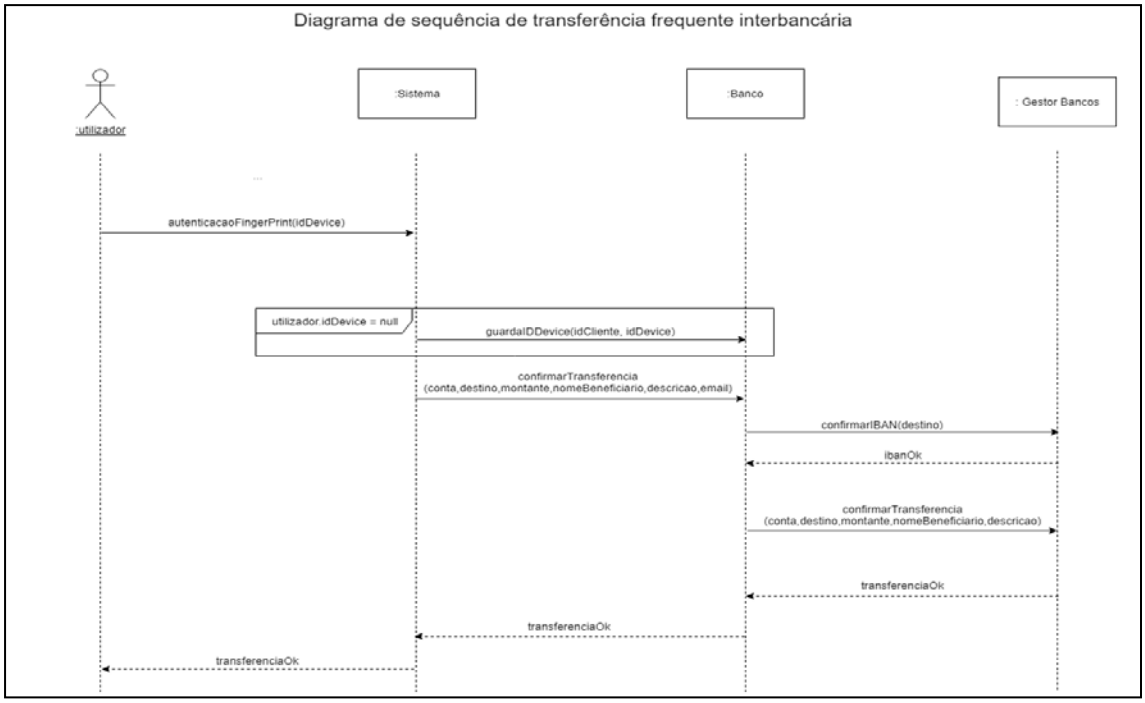
(https://gitlab.com/Henry15/Imagens_da_tese/tree/master/Diagramas_de_sequencia/TransferenciaInter/Benefeciario)

B.7 – UC7: Transferência frequente interbancária

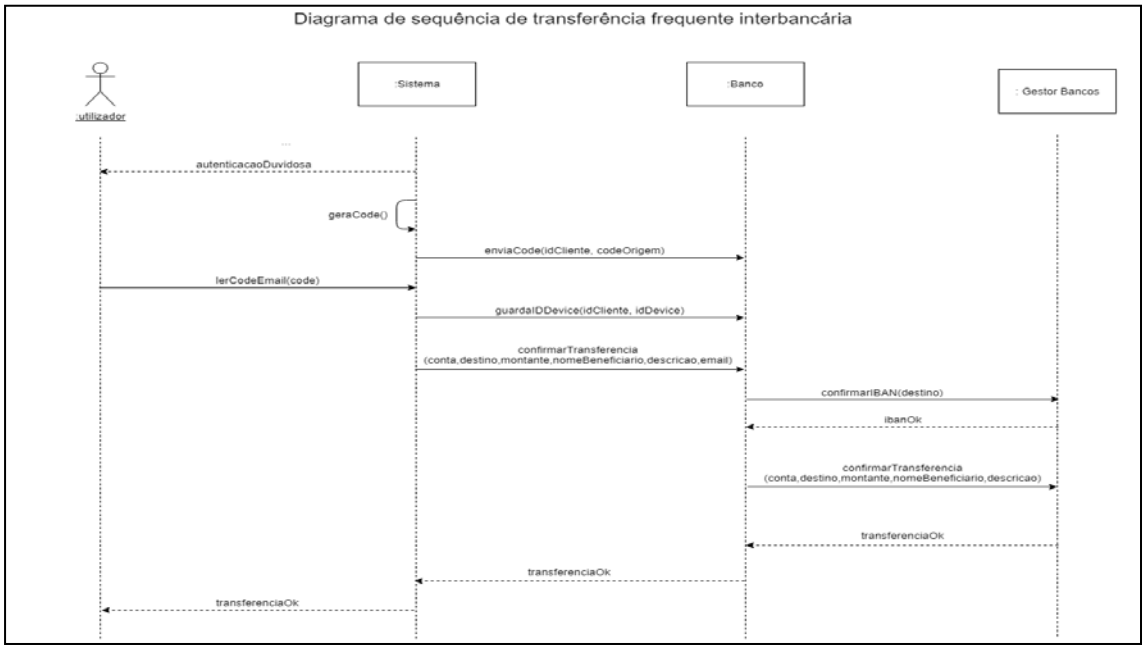
Realização de uma transferência interbancária já efetuada anteriormente, ordenada por ordem de frequência, utilizando as quatro posições do código pessoal como processo de autenticação.



Extensão da realização de uma transferência interbancária já efetuada anteriormente, ordenada por ordem de frequência, utilizando a impressão digital como processo de autenticação.



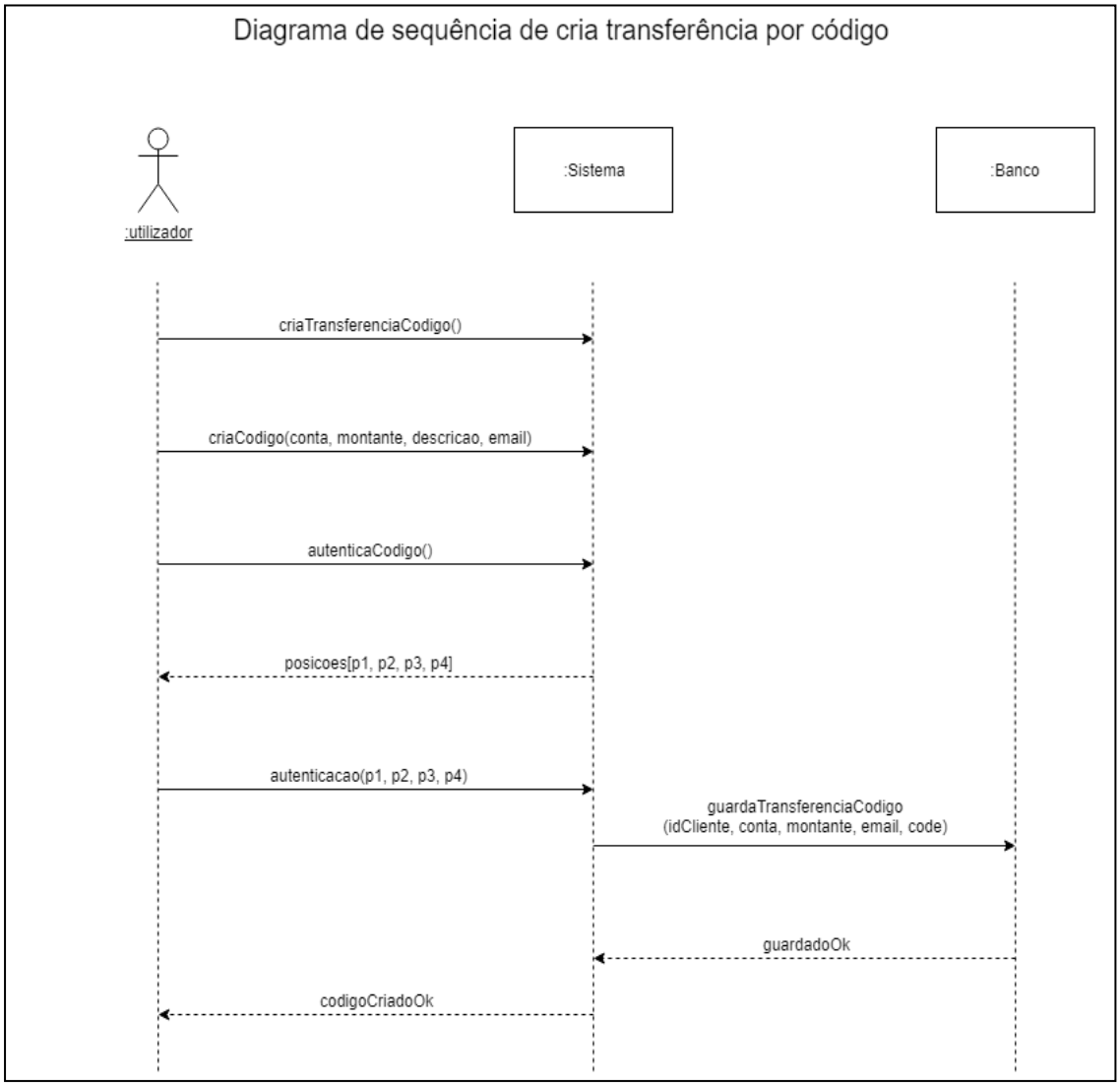
Extensão da realização de uma transferência interbancária já efetuada anteriormente, ordenada por ordem de frequência, quando o sistema deteta uma autenticação duvidosa.



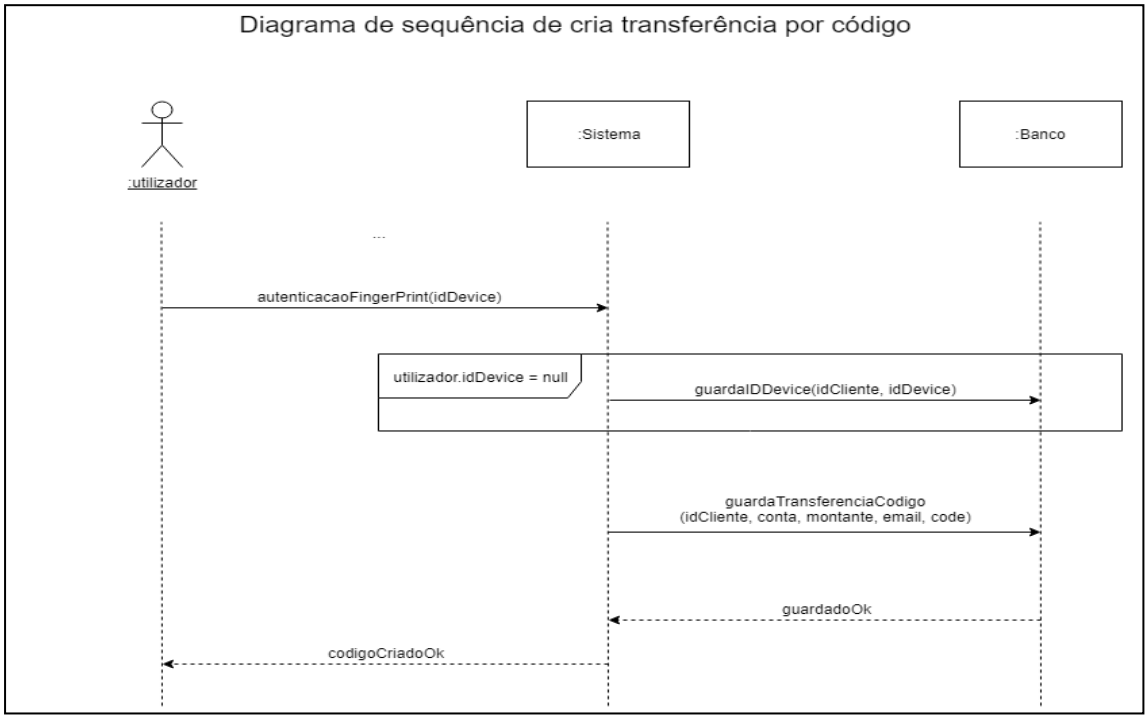
(https://gitlab.com/Henry15/Imagens_da_tese/tree/master/Diagramas_de_sequencia/TransferenciaInter/Frequente)

B.8 – UC8: Cria Transferência por código

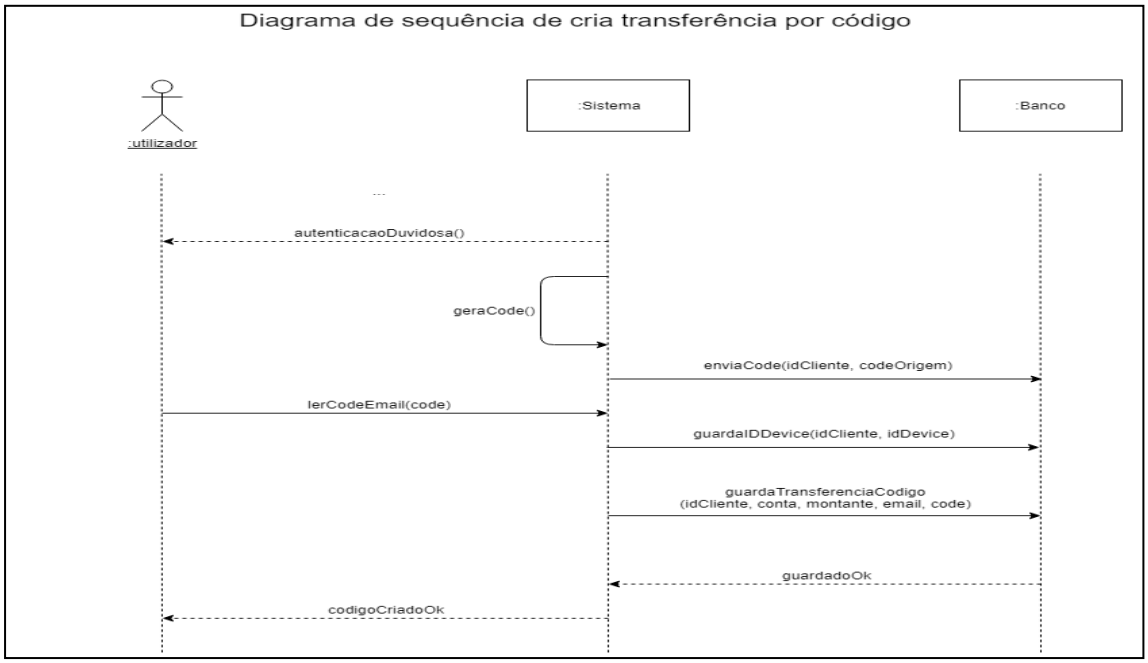
Criação de um código que permite a realização de uma transferência utilizando as quatro posições do código pessoal como processo de autenticação.



Extensão para criação de um código que permite a realização de uma transferência utilizando a impressão digital como processo de autenticação.



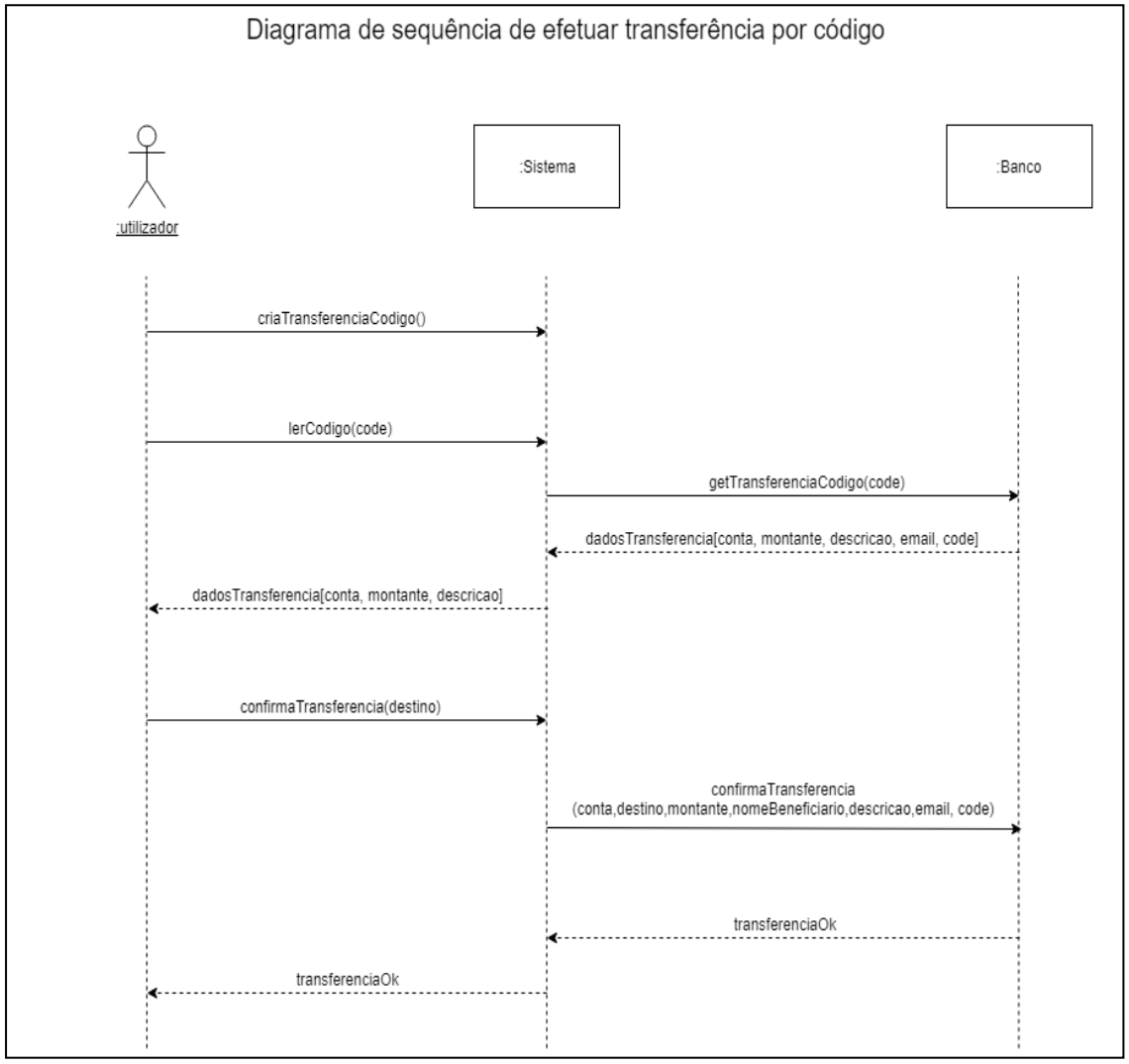
Extensão para criação de um código que permite a realização de uma transferência quando o sistema deteta uma autenticação duvidosa.



(https://gitlab.com/Henry15/Imagens_da_tese/tree/master/Diagramas_de_sequencia/Codigo/Cria)

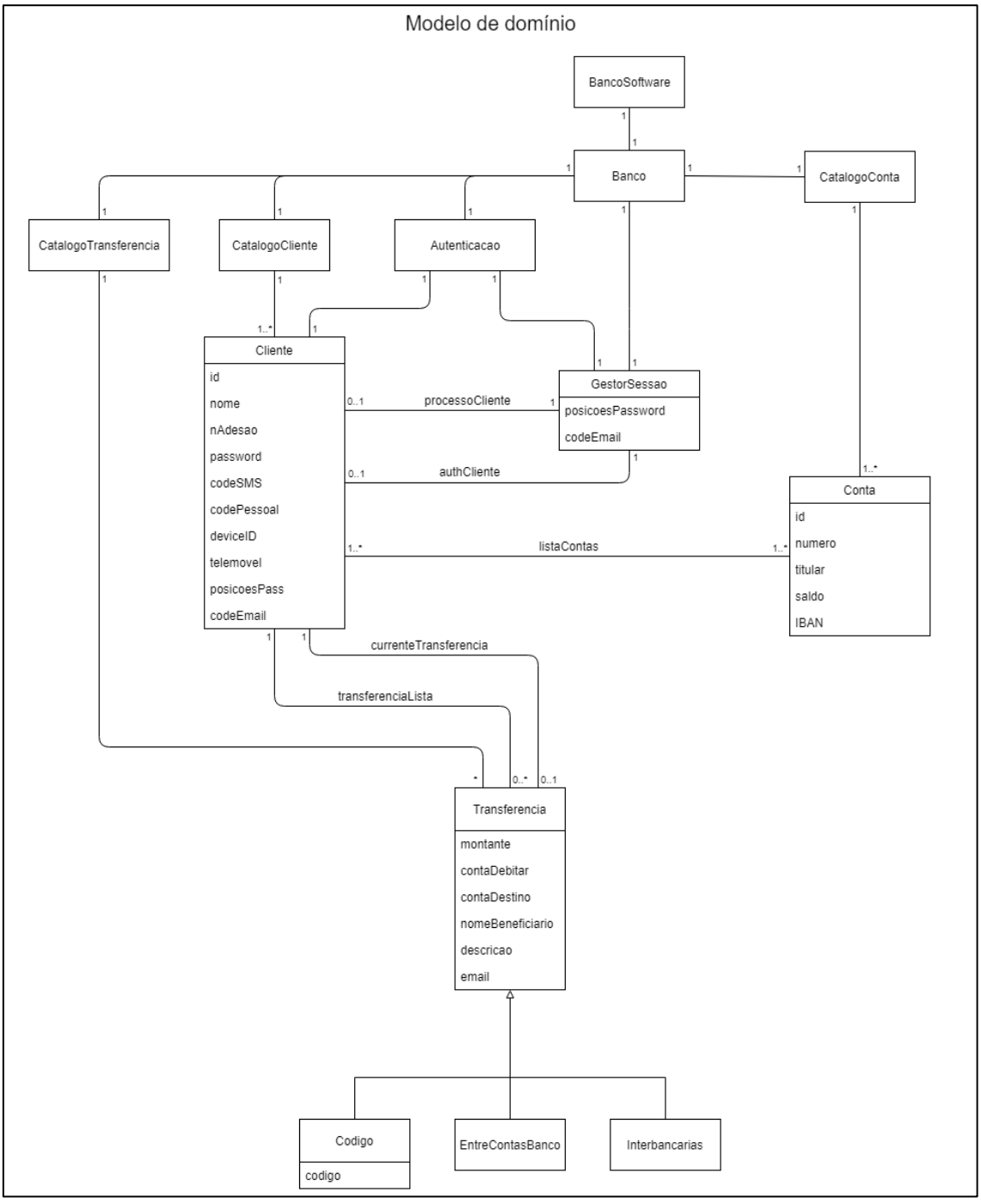
B.9 – UC9: Efetua Transferência por código

Realização de uma transferência através de um código previamente criado.



(https://gitlab.com/Henry15/Imagens_da_tese/tree/master/Diagramas_de_sequencia/Codigo/Efetua)

Anexo C – Modelo de domínio



(https://gitlab.com/Henry15/Imagens_da_tese/blob/master/Modelo_de_dominio/Modelo_dominio.png)

Anexo D – Contratos das operações

Contrato: login(nAdesao, password)

Cross References: UC1 "Login de utilizador"

Pré-condições:

- Há uma instância gs de GestorSessao criada.
- Há uma instância ct de CatalogoTransferencia criada.
- Há uma instância auth de Autenticacao criada.
- Há uma instância catalogoConta de CatalogoConta
- Há uma associação formada entre o gs e o ct.
- Há uma associação formada entre o gs e o catalogoConta.
- Há uma associação formada entre o gs e o auth.

Pós-condições:

- Uma instância processoCliente de Cliente é criada (instância criada).
- Há a iniciação do atributo id da instância processoCliente de Cliente (atributo criado).
- Há a iniciação do atributo nome da instância processoCliente de Cliente (atributo criado).
- Há a iniciação do atributo nAdesao da instância processoCliente de Cliente (atributo criado).
- Há a iniciação do atributo password da instância processoCliente de Cliente (atributo criado).
- Há a iniciação do atributo telemovel da instância processoCliente de Cliente (atributo criado).
- A instância processoCliente é associada ao gs de GestorSessao (associação formada).
- Há uma associação formada entre o processoCliente e o ct (associação formada).
- Há uma associação formada entre o processoCliente e o auth (associação formada).
- Há uma associação formada entre o processoCliente e o catalogoConta (associação formada).

Parâmetros:

- **nAdesao:** Número de nove dígitos atribuído ao utilizador quando pretende aceder aos seus dados financeiros através da plataforma web.
- **password:** Password numérica de nove dígitos atribuída ao utilizador quando deseja utilizar a plataforma mobile. De possível alteração.

Contrato: numeroTelemovel(telemovel, idDevice)

Cross References: UC1 "Login de utilizador"

Pré-condições:

- Há uma instância gs de GestorSessao criada.
- Há uma instância processoCliente de Cliente criada.
- Há uma associação formada entre o processoCliente e o gs.

Pós-condições:

- O atributo idDevice da instância processoCliente torna-se idDevice (atributo criado).
- O atributo codeSMS da instância processoCliente torna-se codeSMS (atributo criado).

Parâmetros:

- **telemovel:** Número de telemóvel do utilizador. A ser utilizado no envio de SMS.
- **idDevice:** Identificador do dispositivo hardware (telemóvel). Corresponde a um valor atribuído pelo fabricante e permite distinguir os diversos equipamentos no mercado. Utilizado para associar o equipamento hardware ao utilizador, de modo a detetar futuras autenticações duvidosas.

Contrato: codigoPessoal(codigoSMS, codigoPessoal)

Cross References: UC1 "Login de utilizador"

Pré-condições:

- Há uma instância gs de GestorSessao criada.
- Há uma instância processoCliente de Cliente criada.
- Há uma associação formada entre o processoCliente e o gs.

Pós-condições:

- O atributo codePessoal da instância processoCliente torna-se codigoPessoal (atributo criado).

Parâmetros:

- **codigoSMS:** Código criado pela aplicação e que será enviado para o utilizador através de um SMS. Garante que o número indicado corresponde efetivamente ao utilizador.
- **codigoPessoal:** Número de seis dígitos escolhido pelo utilizador. Será utilizado no processo de autenticação por código pessoal.

Contrato: autenticaCodigo()

Cross References: UC1 "Login de utilizador"

Pré-condições:

- Há uma instância gs de GestorSessao criada.
- Há uma instância processoCliente de Cliente criada.
- Há uma associação formada entre o processoCliente e o gs.

Pós-condições:

- Há a iniciação do atributo posicoesPassword da instância gs de GestorSessao (atributo criado).

Contrato: autenticacao(p1, p2, p3, p4)

Cross References: UC1 "Login de utilizador"

Pré-condições:

- Há uma instância gs de GestorSessao criada.
- Há uma instância processoCliente de Cliente criada.
- Há uma associação formada entre o processoCliente e o gs.

Pós-condições:

- Uma instância authCliente de Cliente é criada igual ao processoCliente (instância criada).
- Há uma associação entre o gs de GestorSessao e o authCliente de Cliente (associação formada).
- Existe uma desassociação entre a instância processoCliente de Cliente e a instância gs de GestorSessao.

Parâmetros:

- **p1, p2, p3, p4:** Correspondem aos dígitos do código pessoal do utilizador, nas posições pedidas pelo sistema. O sistema irá pedir que o utilizador introduza os dígitos de quatro posições diferentes do seu código pessoal. Acontece sempre quando tenta entrar na sua conta ou realiza transações.

Contrato: autenticacaoFingerPrint(idDevice)

Cross References: UC1 "Login de utilizador"

Pré-condições:

- Há uma instância gs de GestorSessao criada.
- Há uma instância processoCliente de Cliente criada.
- Há uma associação formada entre o processoCliente e o gs.

Pós-condições:

- Uma instância authCliente de Cliente é criada igual ao processoCliente (instância criada).
- Há uma associação entre o gs de GestorSessao e o authCliente de Cliente (associação formada).
- Existe uma desassociação entre a instância processoCliente de Cliente e a instância gs de GestorSessao.

Parâmetros:

- **idDevice:** Identificador do dispositivo hardware (telemóvel). Corresponde a um valor atribuído pelo fabricante e permite distinguir os diversos equipamentos no mercado. Utilizado para associar o equipamento hardware ao utilizador, de modo a detetar futuras autenticações duvidosas.

Contrato: geraCode()

Cross References: UC1 "Login de utilizador"

Pré-condições:

- Há uma instância gs de GestorSessao criada.
- Há uma instância processoCliente de Cliente criada.
- Há uma associação formada entre o processoCliente e o gs.

Pós-condições:

- Inicialização do atributo codeOrigem de gs (atributo criado).

Contrato: lerCodeEmail(code)

Cross References: UC1 "Login de utilizador"

Pré-condições:

- Há uma instância gs de GestorSessao criada.
- Há uma instância processoCliente de Cliente criada.
- Há uma associação formada entre o processoCliente e o gs.

Pós-condições:

- Uma instância authCliente de Cliente é criada igual ao processoCliente (instância criada).
- Há uma associação entre o gs de GestorSessao e o authCliente de Cliente (associação formada).
- Existe uma desassociação entre a instância processoCliente de Cliente e a instância gs de GestorSessao.

Parâmetros:

- **code:** Número de nove dígitos enviado para o email do utilizador. Sempre que o sistema identifica uma autenticação duvidosa cria um número de nove dígitos e envia para o email do utilizador. Só é possível realizar a tarefa se o número enviado para o email for igual ao parâmetro de entrada.

Contrato: escolheTransferenciaEntreContasBanco()

Cross References: UC2 "Transferência entre contas do mesmo banco", **UC4 "**Transferência beneficiário entre contas do mesmo banco", **UC5 "**Transferência frequente entre contas do mesmo banco".

Pré-condições:

- Há uma instância authCliente de Cliente.

Pós-condições:

- Uma instância de currenteTransferencia de EntreContasBanco é criada (instância criada).
- Há uma associação entre o authCliente de Cliente e a currenteTransferencia (associação formada).

Contrato: mostraContasCliente()

Cross References: UC2 "Transferência entre contas do mesmo banco", **UC3 "**Transferência interbancária".

Pré-condições:

- Há uma instância authCliente de Cliente.

Pós-condições:

- Uma instância de listaContas de Conta é criada (instância criada).
- Há uma associação entre o authCliente de Cliente e a listaContas (associação formada).

Contrato: confirmaTransferencia(conta, destino, montante, nomeBeneficiario, descricao, email)

Cross References: UC2 "Transferência entre contas do mesmo banco", **UC3 "**Transferência interbancária".

Pré-condições:

- Há uma instância currenteTransferencia de Transferencia.
- Há uma instância authCliente de Cliente.

Pós-condições:

- O atributo contaDebitar da instância currenteTransferencia torna-se conta (atributo criado).
- O atributo contaDestino da instância currenteTransferencia torna-se destino (atributo criado).
- O atributo montante da instância currenteTransferencia torna-se montante (atributo criado).
- O atributo nomeBeneficiario da instância currenteTransferencia torna-se nomeBeneficiario (atributo criado).
- O atributo descricao da instância currenteTransferencia torna-se descricao (atributo criado).
- O atributo email da instância currenteTransferencia torna-se email (atributo criado).

Parâmetros:

- **conta:** Número da conta onde vai ser debitado o valor da transação.
- **destino:** Número ou IBAN da conta onde vai ser depositado o valor da transação.
- **montante:** Valor da transação a ser realizada entre duas contas.
- **nomeBeneficiario:** Nome do titular da conta de destino da transação.
- **descricao:** Uma breve justificação sobre a realização da transação.
- **email:** Email do titular da conta de debito. Utilizado para enviar a notificação da realização da transferência e o seu comprovativo.

Contrato: autenticaCodigo()

Cross References: UC2 "Transferência entre contas do mesmo banco", **UC4 "**Transferência beneficiário entre contas do mesmo banco", **UC5 "**Transferência frequente entre contas do mesmo banco", **UC3 "**Transferência interbancária", **UC6 "**Transferência beneficiário interbancária", **UC7 "**Transferência frequente interbancária", **UC8** "Cria Transferência por código".

Pré-condições:

- Há uma instância currenteTransferencia de Transferencia.
- Há uma instância authCliente de Cliente.

Pós-condições:

- O atributo posicoesPass da instância authCliente é inicializado (atributo criado).

Contrato: autenticacao(p1, p2, p3, p4)

Cross References: UC2 "Transferência entre contas do mesmo banco", **UC4 "**Transferência beneficiário entre contas do mesmo banco", **UC5 "**Transferência frequente entre contas do mesmo banco", **UC3 "**Transferência interbancária", **UC6 "**Transferência beneficiário interbancária", **UC7 "**Transferência frequente interbancária", **UC8** "Cria Transferência por código".

Pré-condições:

- Há uma instância currenteTransferencia de Transferencia.
- Há uma instância authCliente de Cliente.

Pós-condições:

- A currenteTransferencia é concluída e finalizada.
- Existe a desassociação entre o authCliente e a currentTransferencia.

Parâmetros:

- **p1, p2, p3, p4:** Correspondem aos dígitos do código pessoal do utilizador, nas posições pedidas pelo sistema. O sistema irá pedir que o utilizador introduza os dígitos de quatro posições diferentes do seu código pessoal. Acontece sempre quando tenta entrar na sua conta ou realiza transações.

Contrato: autenticacaoFingerPrint(idDevice)

Cross References: UC2 "Transferência entre contas do mesmo banco", **UC4 "**Transferência beneficiário entre contas do mesmo banco", **UC5 "**Transferência frequente entre contas do mesmo banco", **UC3 "**Transferência interbancária", **UC6 "**Transferência beneficiário interbancária", **UC7 "**Transferência frequente interbancária", **UC8** "Cria Transferência por código".

Pré-condições:

- Há uma instância currenteTransferencia de Transferencia.
- Há uma instância authCliente de Cliente.

Pós-condições:

- A currenteTransferencia é concluída e finalizada.

- Existe a desassociação entre o authCliente e a currentTransferencia.

Parâmetros:

- **idDevice:** Identificador do dispositivo hardware (telemóvel). Corresponde a um valor atribuído pelo fabricante e permite distinguir os diversos equipamentos no mercado. Utilizado para associar o equipamento hardware ao utilizador, de modo a detetar futuras autenticações duvidosas.

Contrato: geraCode()

Cross References: UC2 "Transferência entre contas do mesmo banco", **UC4 "**Transferência beneficiário entre contas do mesmo banco", **UC5 "**Transferência frequente entre contas do mesmo banco", **UC3 "**Transferência interbancária", **UC6 "**Transferência beneficiário interbancária", **UC7 "**Transferência frequente interbancária", **UC8** "Cria Transferência por código".

Pré-condições:

- Há uma instância currenteTransferencia de Transferencia.
- Há uma instância authCliente de Cliente.

Pós-condições:

- Inicialização do atributo codeEmail da instância authCliente (atributo criado).

Contrato: lerCodeEmail(code)

Cross References: UC2 "Transferência entre contas do mesmo banco", **UC4 "**Transferência beneficiário entre contas do mesmo banco", **UC5 "**Transferência frequente entre contas do mesmo banco", **UC3 "**Transferência interbancária", **UC6 "**Transferência beneficiário interbancária", **UC7 "**Transferência frequente interbancária", **UC8** "Cria Transferência por código".

Pré-condições:

- Há uma instância currenteTransferencia de Transferencia.
- Há uma instância authCliente de Cliente.

Pós-condições:

- A currenteTransferencia é concluída e finalizada.
- Existe a desassociação entre o authCliente e a currentTransferencia.

Parâmetros:

- **code:** Número de nove dígitos enviado para o email do utilizador. Sempre que o sistema identifica uma autenticação duvidosa cria um número de nove dígitos e envia para o email do utilizador. Só é possível realizar a tarefa se o número enviado para o email for igual ao parâmetro de entrada.

Contrato: escolheTransferenciaInterbancarias()

Cross References: UC3 "Transferência interbancária", **UC6 "**Transferência beneficiário interbancária", **UC7 "**Transferência frequente interbancária".

Pré-condições:

- Há uma instância authCliente de Cliente.

Pós-condições:

- Uma instância de currenteTransferencia de Interbancarias é criada (instância criada).
- Há uma associação entre o authCliente de Cliente e a currenteTransferencia (associação formada).

Contrato: getTransferenciasBenefeciarioECB()

Cross References: UC4 "Transferência beneficiário entre contas do mesmo banco".

Pré-condições:

- Há uma instância authCliente de Cliente.

Pós-condições:

- Uma instância de transferenciaLista de Transferencia é criada (instância criada).
- Há uma associação entre o authCliente de Cliente e a transferenciaLista (associação formada).

Contrato: getTransferenciasFrequentesECB()

Cross References: UC5 "Transferência frequente entre contas do mesmo banco".

Pré-condições:

- Há uma instância authCliente de Cliente.

Pós-condições:

- Uma instância de transferenciaLista de Transferencia é criada (instância criada).
- Há uma associação entre o authCliente de Cliente e a transferenciaLista (associação formada).

Contrato: getTransferenciasBeneficiarioI()

Cross References: UC6 "Transferência beneficiário interbancária".

Pré-condições:

- Há uma instância authCliente de Cliente.

Pós-condições:

- Uma instância de transferenciaLista de Transferencia é criada (instância criada).
- Há uma associação entre o authCliente de Cliente e a transferenciaLista (associação formada).

Contrato: getTransferenciasFrequentesI()

Cross References: UC7 "Transferência frequente interbancária".

Pré-condições:

- Há uma instância authCliente de Cliente.

Pós-condições:

- Uma instância de transferenciaLista de Transferencia é criada (instância criada).
- Há uma associação entre o authCliente de Cliente e a transferenciaLista (associação formada).

Contrato: confirmaTransferencia(transferencia)

Cross References: UC4 "Transferência beneficiário entre contas do mesmo banco", UC5 "Transferência frequente entre contas do mesmo banco", **UC6** "Transferência beneficiário interbancária", **UC7** "Transferência frequente interbancária".

Pré-condições:

- Há uma instância authCliente de Cliente.
- Já existe uma Transferencia t.

Pós-condições:

- É instanciada a currenteTransferencia igual à Transferencia t (instância criada).
- Existe uma associação entre o authCliente e a currenteTransferencia (associação formada).

Parâmetros:

- **transferencia:** Transferência selecionada pelo cliente para ser realizada, após ser exibida um conjunto de transferências já realizadas pelo utilizador anteriormente.

Contrato: criaTransferenciaCodigo()

Cross References: UC8 "Cria Transferência por código", **UC9** "Efetua Transferência por código".

Pré-condições:

- Há uma instância authCliente de Cliente.

Pós-condições:

- Uma instância de currenteTransferencia de Codigo é criada (instância criada).
- Existe uma associação entre o authCliente e a currenteTransferencia (associação formada).

Contrato: criaCodigo(conta, montante, descricao, email)

Cross References: UC8 "Cria Transferência por código".

Pré-condições:

- Há uma instância authCliente de Cliente.
- Existe uma instância currenteTransferencia de Codigo.
- Existe uma associação entre authCliente e currenteTransferencia.

Pós-condições:

- O atributo contaDebitar da instância currenteTransferencia torna-se conta (atributo criado).

- O atributo montante da instância currenteTransferencia torna-se montante (atributo criado).
- O atributo descricao da instância currenteTransferencia torna-se descricao (atributo criado).
- O atributo email da instância currenteTransferencia torna-se email (atributo criado).
- O atributo codigo da instância currenteTransferencia é inicializado (atributo criado).
- O código da transferência é criado.

Parâmetros:

- **conta:** Número da conta onde vai ser debitado o valor da transação.
- **montante:** Valor da transação a ser realizada entre duas contas.
- **descricao:** Uma breve justificação sobre a realização da transação.
- **email:** Email do titular da conta de debito. Utilizado para enviar a notificação da realização da transferência e o seu comprovativo.

Contrato: autenticacao(p1, p2, p3, p4)

Cross References: UC8 "Cria Transferência por código".

Pré-condições:

- Há uma instância authCliente de Cliente.
- Existe uma instância currenteTransferencia de Codigo.
- Existe uma associação entre authCliente e currenteTransferencia.

Pós-condições:

- O código da transferência é criado.
- Existe a desassociação entre authCliente e currenteTransferencia.

Parâmetros:

- **p1, p2, p3, p4:** Correspondem aos dígitos do código pessoal do utilizador, nas posições pedidas pelo sistema. O sistema irá pedir que o utilizador introduza os dígitos de quatro posições diferentes do seu código pessoal. Acontece sempre quando tenta entrar na sua conta ou realiza transações.

Contrato: autenticacaoFingerPrint(idDevice)

Cross References: UC8 "Cria Transferência por código".

Pré-condições:

- Há uma instância authCliente de Cliente.
- Existe uma instância currenteTransferencia de Codigo.
- Existe uma associação entre authCliente e currenteTransferencia.

Pós-condições:

- O código da transferência é criado.
- Existe a desassociação entre authCliente e currenteTransferencia.

Parâmetros:

- **idDevice:** Identificador do dispositivo hardware (telemóvel). Corresponde a um valor atribuído pelo fabricante e permite distinguir os diversos equipamentos no mercado. Utilizado para associar o equipamento hardware ao utilizador, de modo a detetar futuras autenticações duvidosas.

Contrato: lerCodeEmail(code)

Cross References: UC8 "Cria Transferência por código".

Pré-condições:

- Há uma instância authCliente de Cliente.
- Existe uma instância currenteTransferencia de Codigo.
- Existe uma associação entre authCliente e currenteTransferencia.

Pós-condições:

- O código da transferência é criado.
- Existe a desassociação entre authCliente e currenteTransferencia.

Parâmetros:

- **code:** Número de nove dígitos enviado para o email do utilizador. Sempre que o sistema identifica uma autenticação duvidosa cria um número de nove dígitos e envia para o email do utilizador. Só é possível realizar a tarefa se o número enviado para o email for igual ao parâmetro de entrada.

Contrato: lerCodigo (code)

Cross References: UC9 "Efetua Transferência por código".

Pré-condições:

- Há uma instância authCliente de Cliente.
- Existe uma instância currenteTransferencia de Codigo.
- Existe uma associação entre authCliente e currenteTransferencia.

Pós-condições:

- O atributo contaDebitar da instância currenteTransferencia torna-se conta (atributo criado).
- O atributo montante da instância currenteTransferencia torna-se montante (atributo criado).
- O atributo descricao da instância currenteTransferencia torna-se descricao (atributo criado).
- O atributo codigo da instância currenteTransferencia torna-se code (atributo criado).
- O atributo email da instância currenteTransferencia torna-se email (atributo criado).

Parâmetros:

- **code:** Código que identifica um barcode. Um barcode é criado por um código único que vai associar os dados da transferência ao barcode.

Contrato: confirmaTransferencia(destino)

Cross References: UC9 "Efetua Transferência por código".

Pré-condições:

- Há uma instância authCliente de Cliente.
- Existe uma instância currenteTransferencia de Codigo.
- Existe uma associação entre authCliente e currenteTransferencia.

Pós-condições:

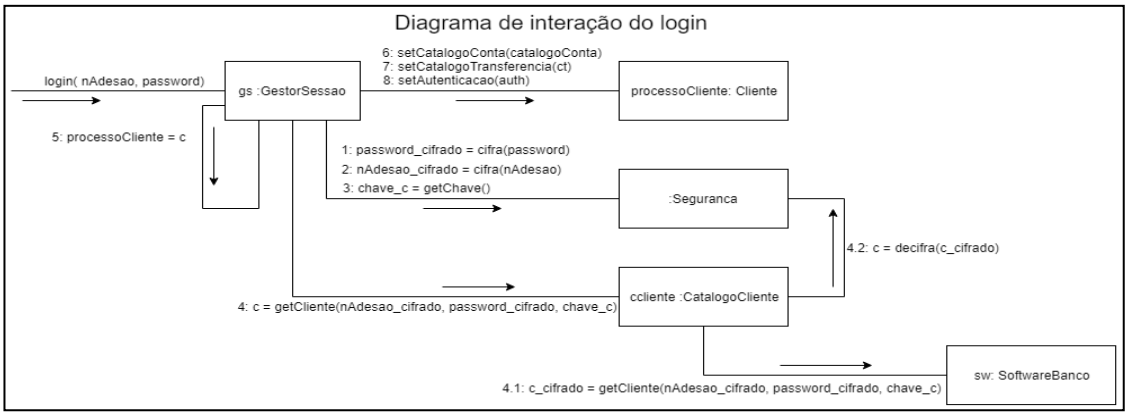
- O atributo contaDestino da instância currenteTransferencia torna-se destino (atributo criado).
- A currenteTransferencia é realizada e finalizada.
- Existe a desassociação entre authCliente e currenteTransferencia.

Parâmetros:

- **destino:** Número da conta onde vai ser depositado o montante da transação.

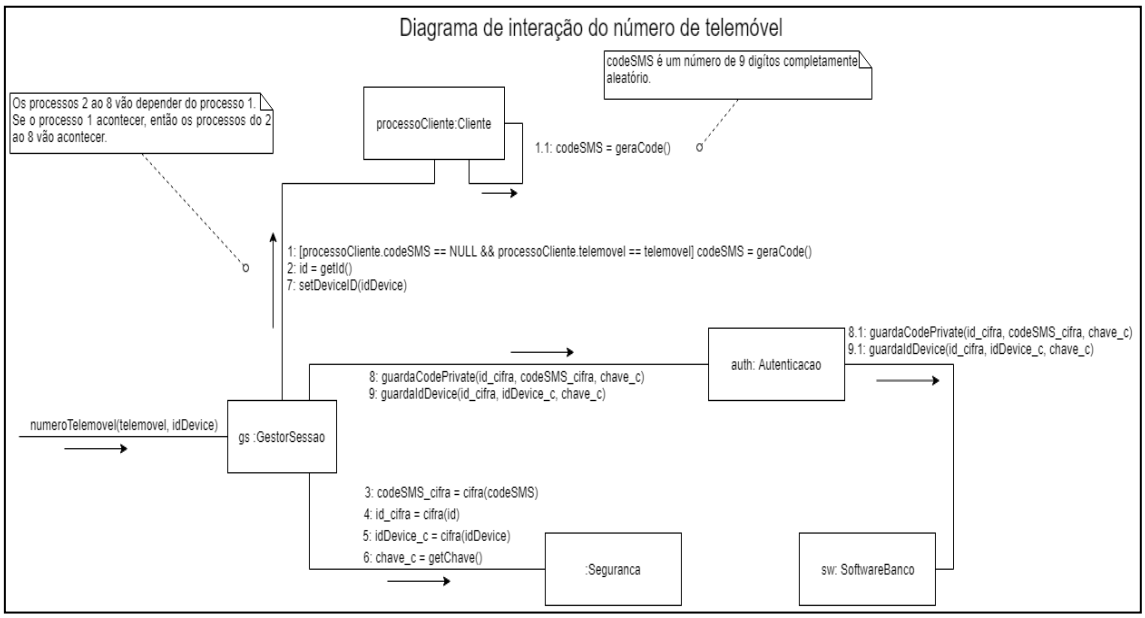
Anexo E – Diagramas de interação

E.1 – login de utilizador



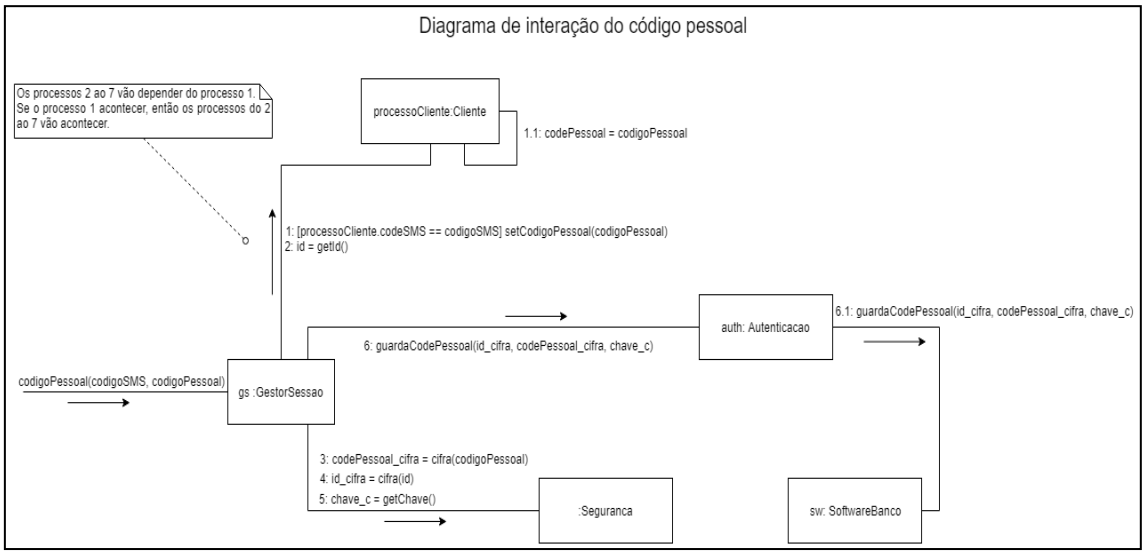
(https://gitlab.com/Henry15/Imagens_da_tese/blob/master/Diagramas_de_interacao/Login/login.png)

E.2 – número de telemóvel



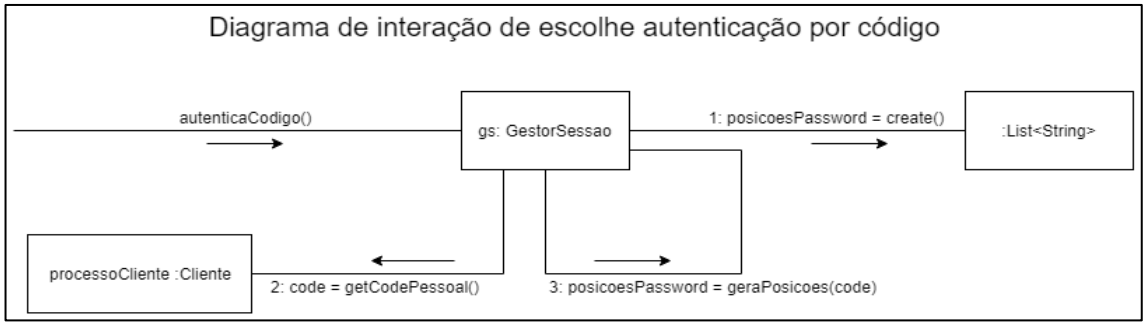
(https://gitlab.com/Henry15/Imagens_da_tese/blob/master/Diagramas_de_interacao/Login/diz_numero_telemovel.png)

E.3 – código pessoal



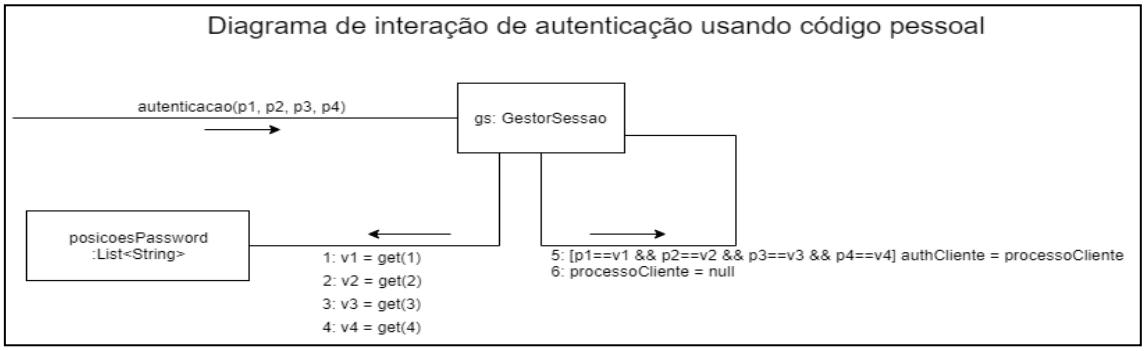
(https://gitlab.com/Henry15/Imagens_da_tese/blob/master/Diagramas_de_interacao/Login/diz_o_codigo_pessoal.png)

E.4 – escolha de posições para autenticação



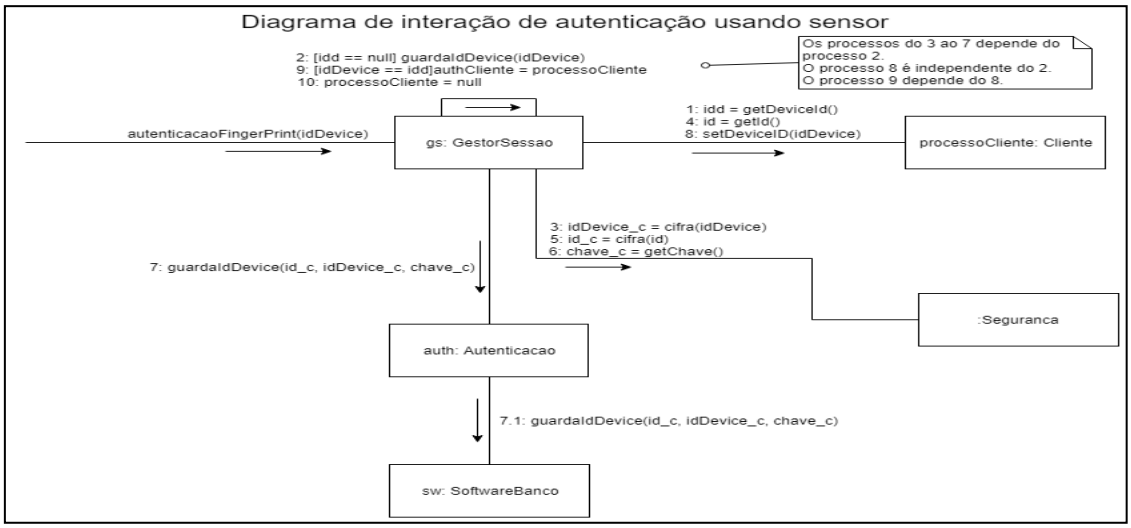
(https://gitlab.com/Henry15/Imagens_da_tese/blob/master/Diagramas_de_interacao/Login/escolhe_autenticacao_por_codigo.png)

E.5 – autenticação usando código pessoal



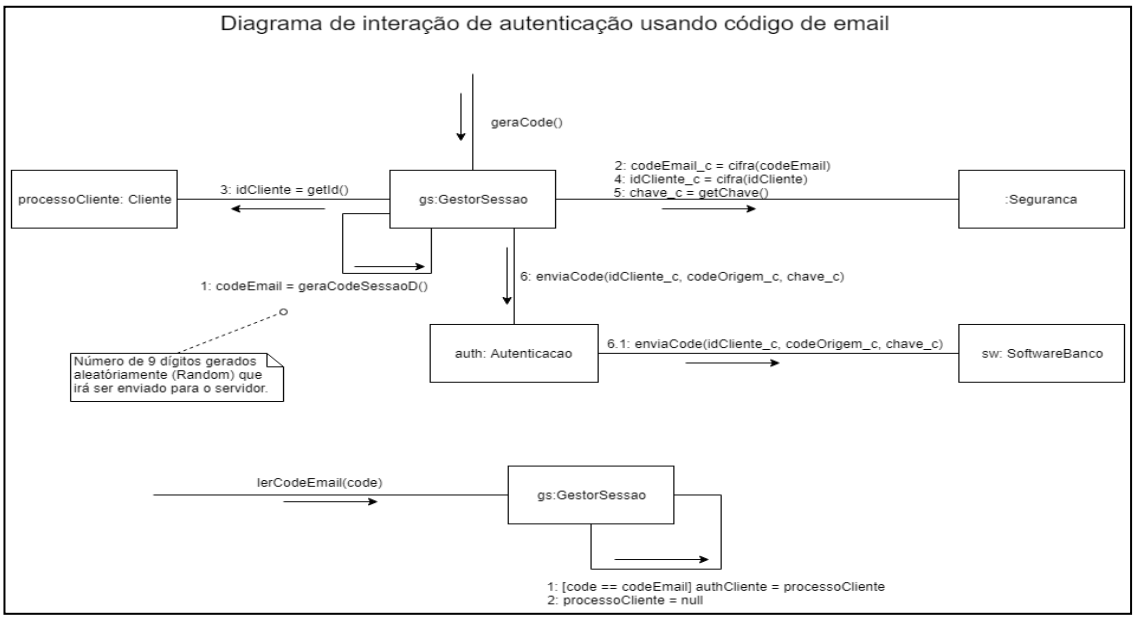
(https://gitlab.com/Henry15/Imagens_da_tese/blob/master/Diagramas_de_interacao/Login/autenticacao_usando_codigo_email.png)

E.6 – autenticação usando sensor



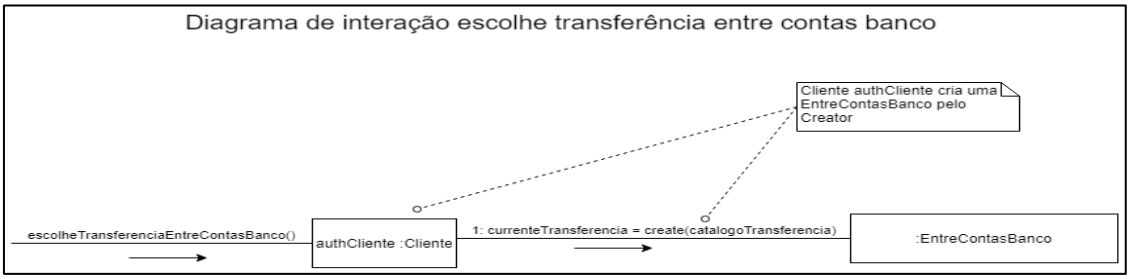
(https://gitlab.com/Henry15/Imagens_da_tese/blob/master/Diagramas_de_interacao/Login/autenticacao_usando_sensor.png)

E.7 – autenticação usando código de email



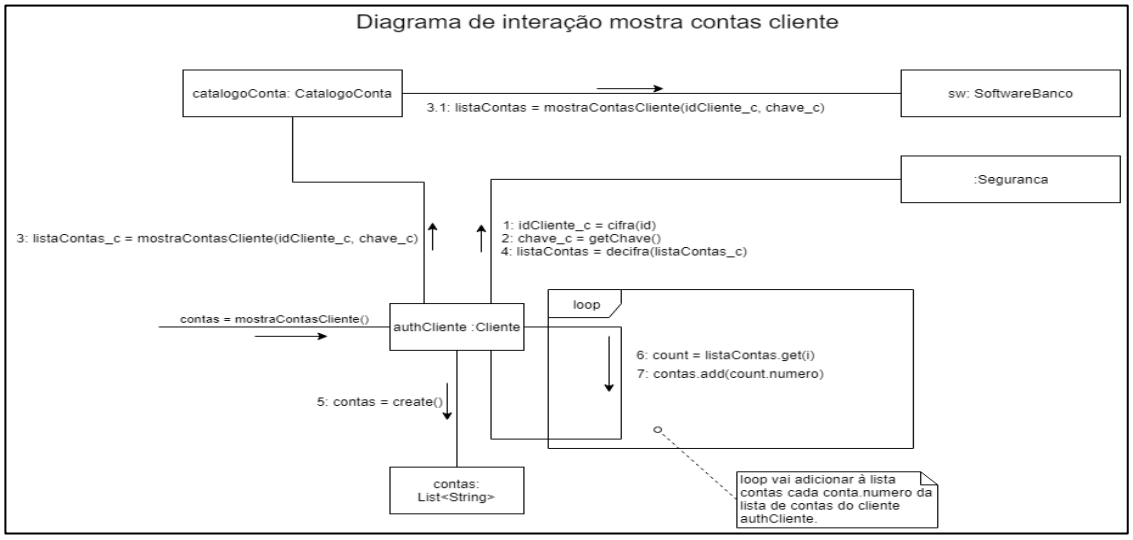
(https://gitlab.com/Henry15/Imagens_da_tese/blob/master/Diagramas_de_interacao/Login/autenticacao_usando_email.png)

E.8 – transferência entre contas do mesmo banco



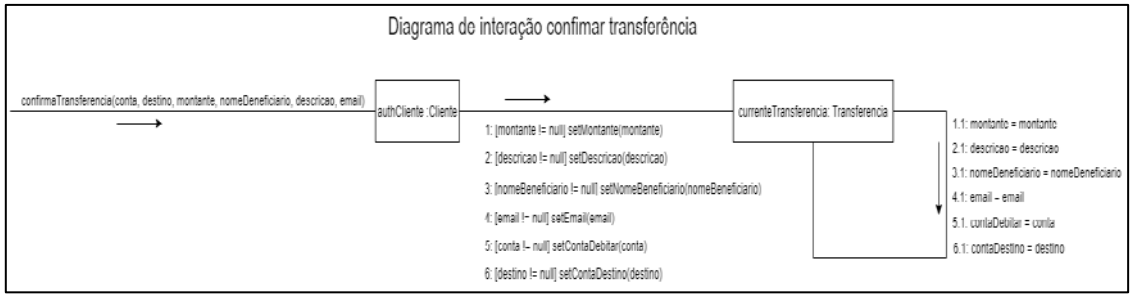
(https://gitlab.com/Henry15/Imagens_da_tese/blob/master/Diagramas_de_interacao/Transferencia/escolhe_transferencia_entre_contas_banco.png)

E.9 – visualização de contas cliente



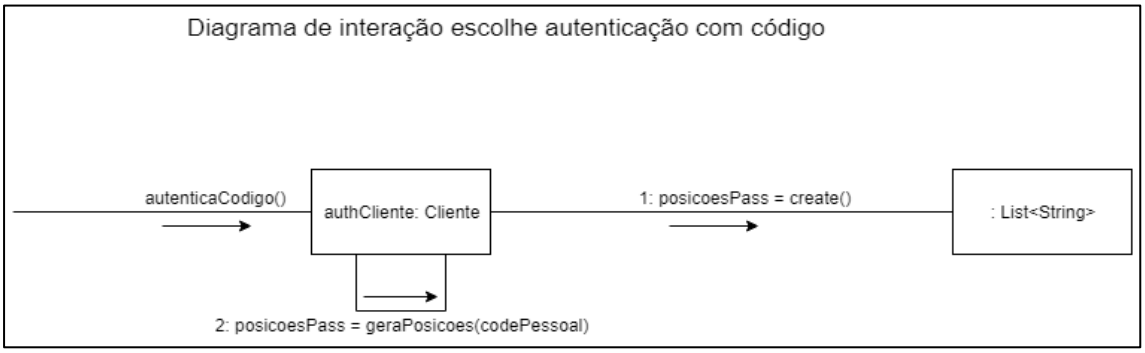
(https://gitlab.com/Henry15/Imagens_da_tese/blob/master/Diagramas_de_interacao/Transferencia/mostra_contas_cliente.png)

E.10 – confirmação de transferência



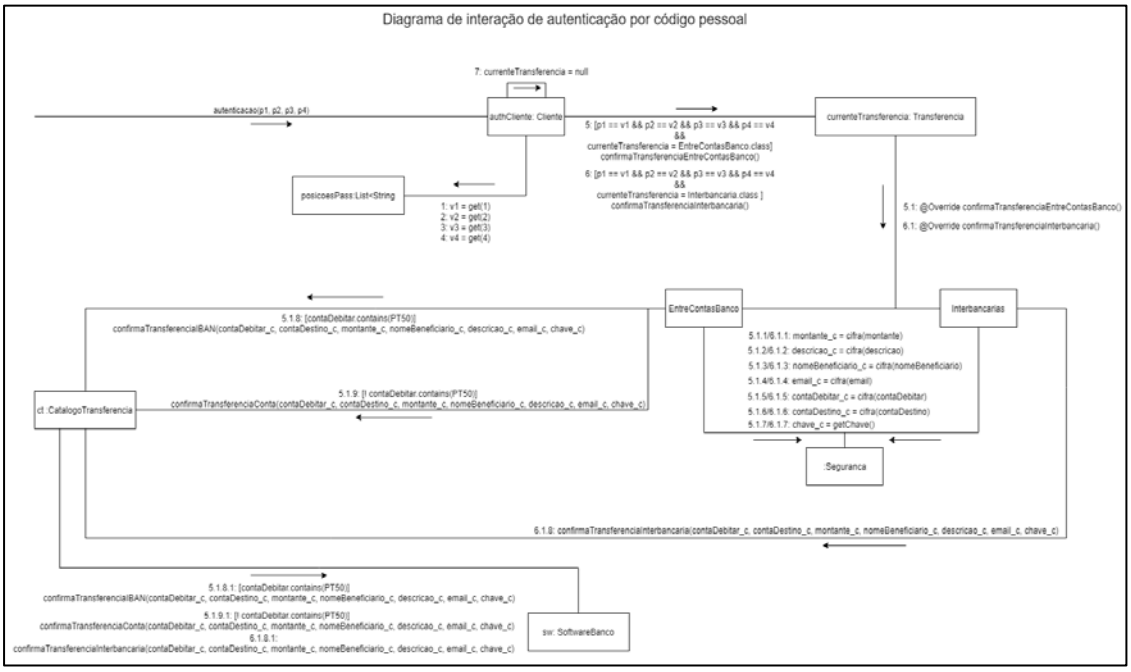
(https://gitlab.com/Henry15/Imagens_da_tese/blob/master/Diagramas_de_interacao/Transferencia/confirma_transferencia_nova.png)

E.11 – escolha de autenticação com código



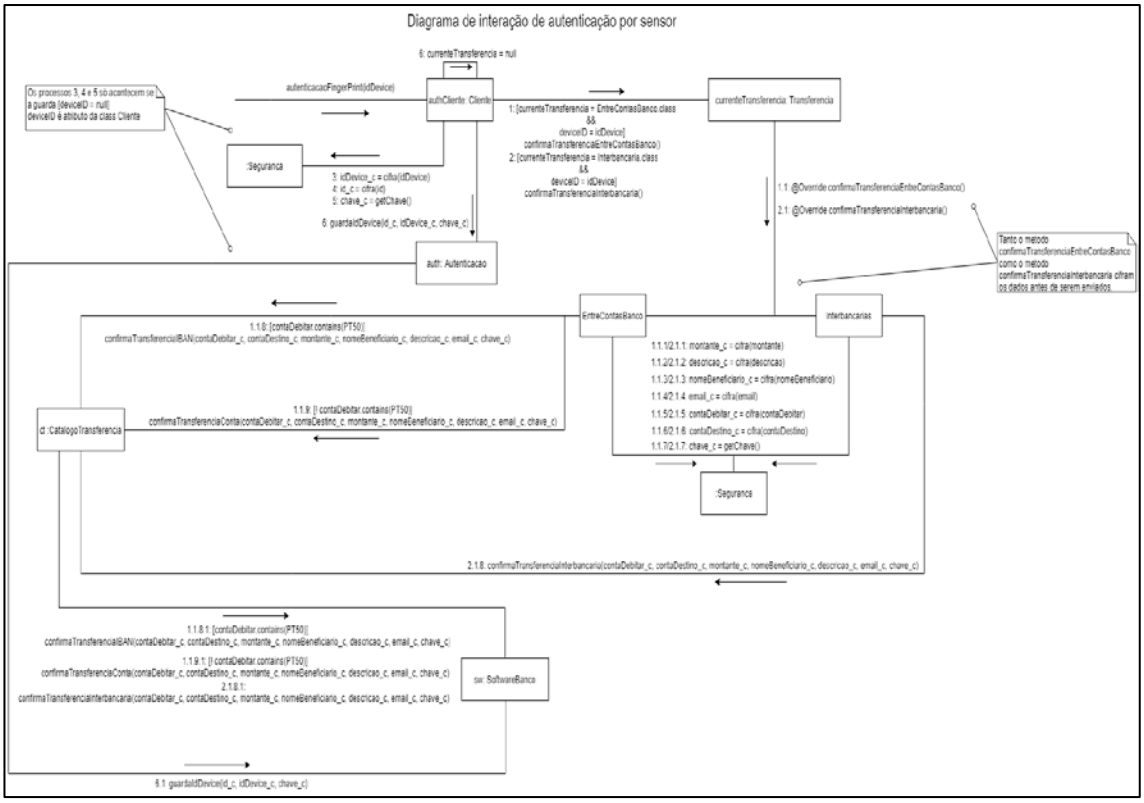
(https://gitlab.com/Henry15/Imagens_da_tese/blob/master/Diagramas_de_interacao/Transferencia/escolhe_autenticacao_com_codigo.png)

E.12 – autenticação com código pessoal



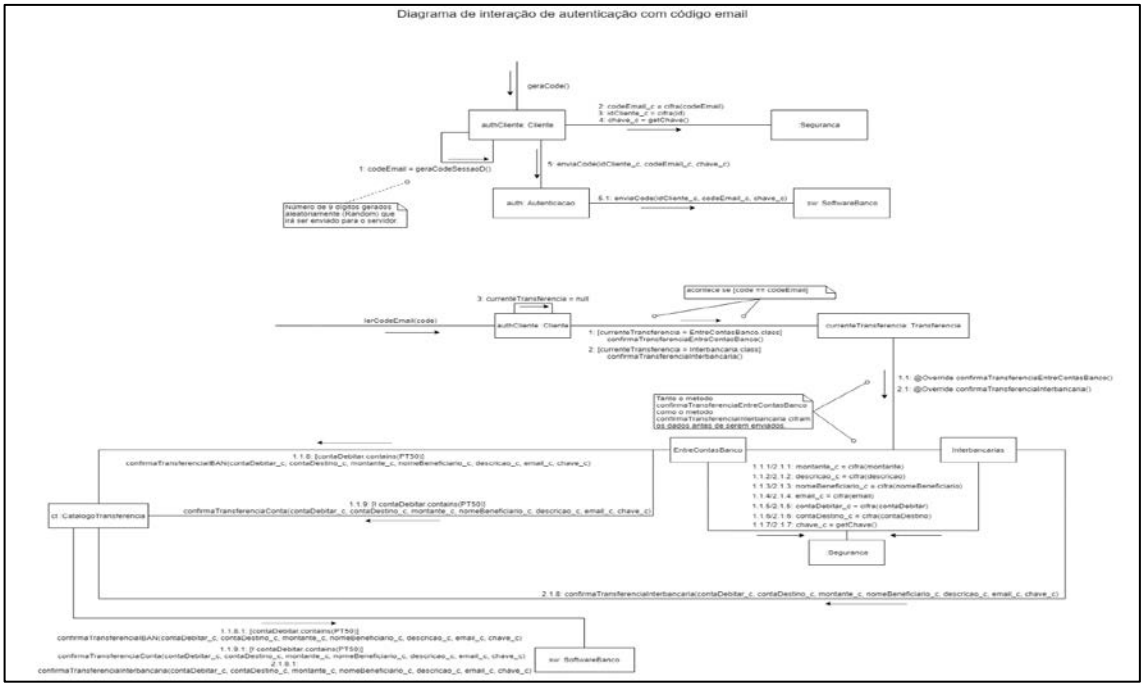
(https://gitlab.com/Henry15/Imagens_da_tese/blob/master/Diagramas_de_interacao/Transferencia/autenticacao_usando_codigo_pessoal.png)

E.13 – autenticação com sensor



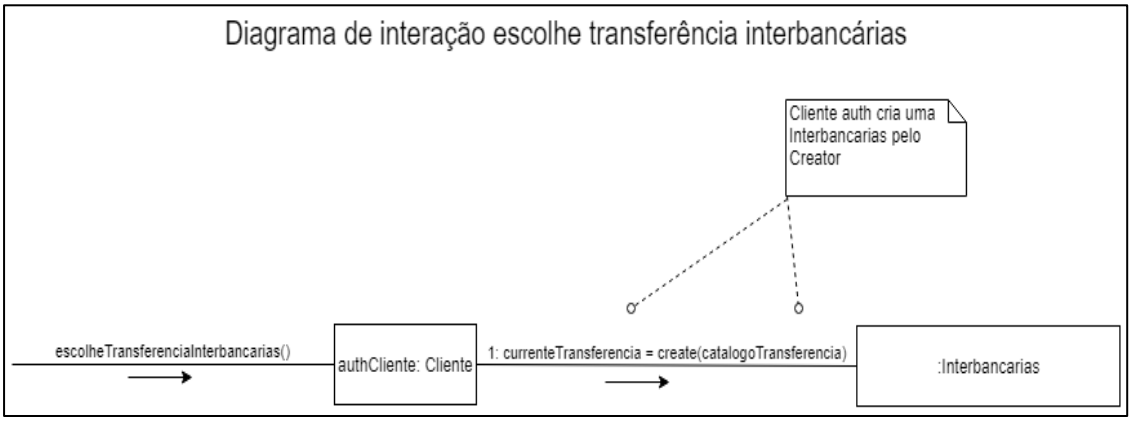
(https://gitlab.com/Henry15/Imagens_da_tese/blob/master/Diagramas_de_interacao/Transferencia/autenticacao_com_sensor.png)

E.14 – autenticação com código de email



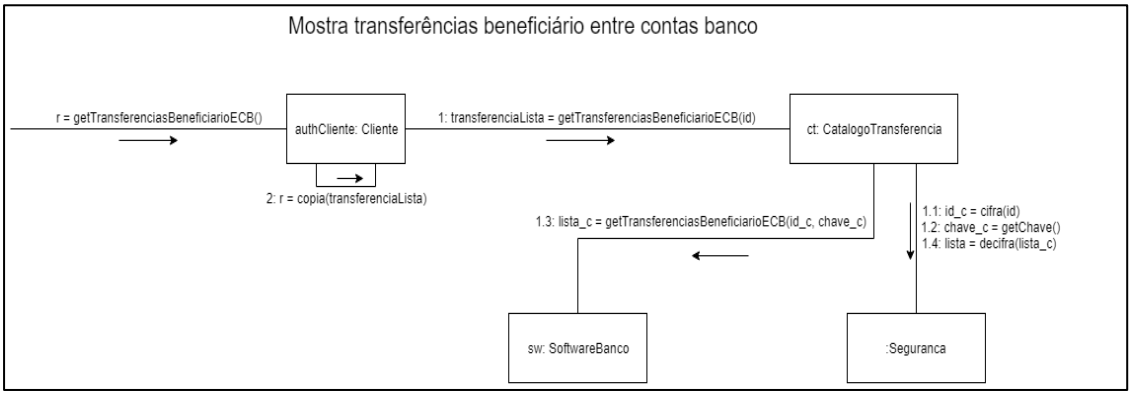
(https://gitlab.com/Henry15/Imagens_da_tese/blob/master/Diagramas_de_interacao/Transferencia/autenticacao_usando_codigo_email.png)

E.15 – escolha de transferência interbancária



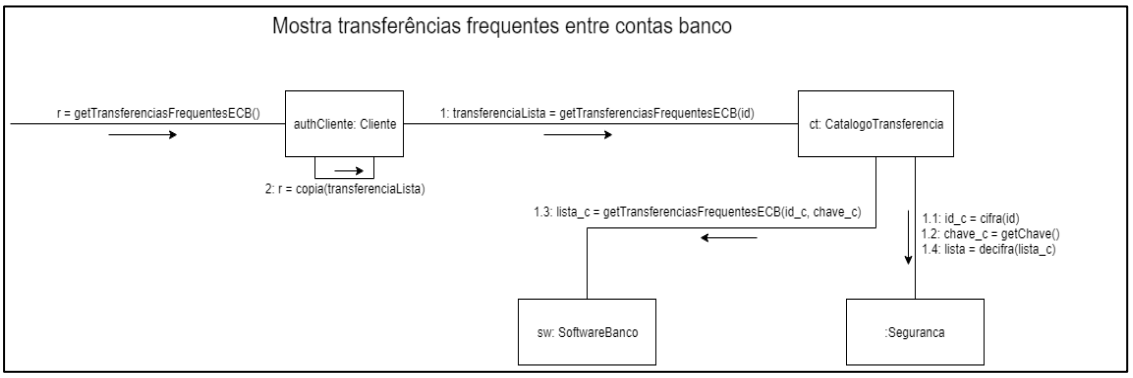
(https://gitlab.com/Henry15/Imagens_da_tese/blob/master/Diagramas_de_interacao/Transferencia/escolhe_transferencias_interbancarias.png)

E.16 – mostra transferências beneficiário entre contas do mesmo banco



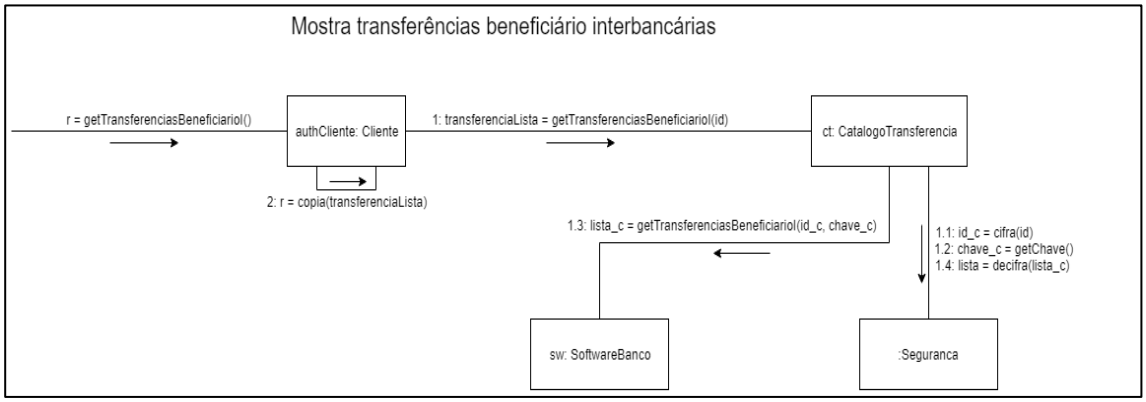
(https://gitlab.com/Henry15/Imagens_da_tese/blob/master/Diagramas_de_interacao/Transferencia/mostra_transferencias_beneficiarioECB.png)

E.17 – mostra transferências frequentes entre contas do mesmo banco



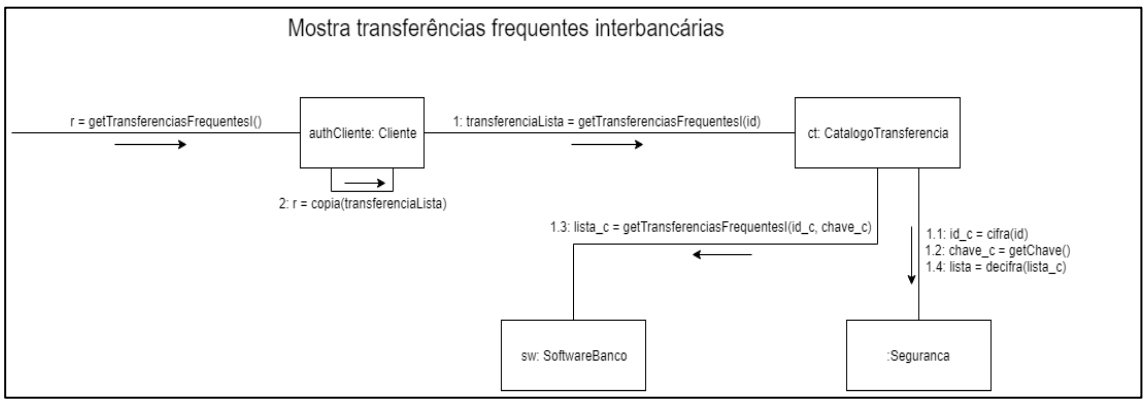
(https://gitlab.com/Henry15/Imagens_da_tese/blob/master/Diagramas_de_interacao/Transferencia/mostra_transferencias_frequentesECB.png)

E.18 – mostra transferências beneficiário interbancárias



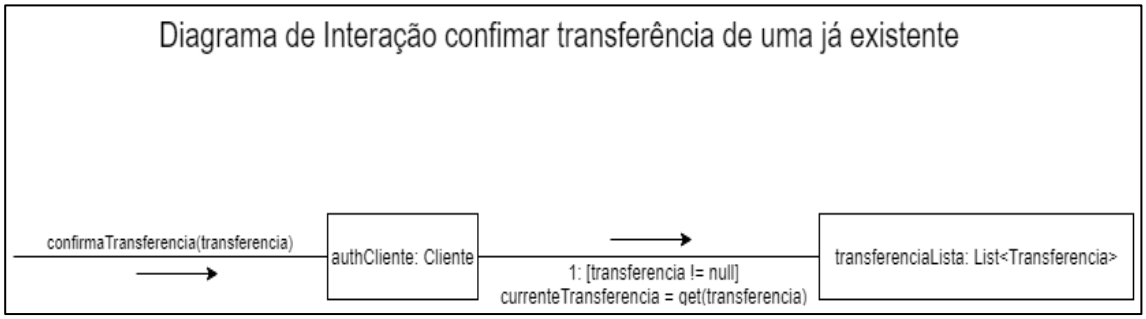
(https://gitlab.com/Henry15/Imagens_da_tese/blob/master/Diagramas_de_interacao/Transferencia/mostra_transferencias_beneficiarioI.png)

E.19 – mostra transferências frequentes interbancárias



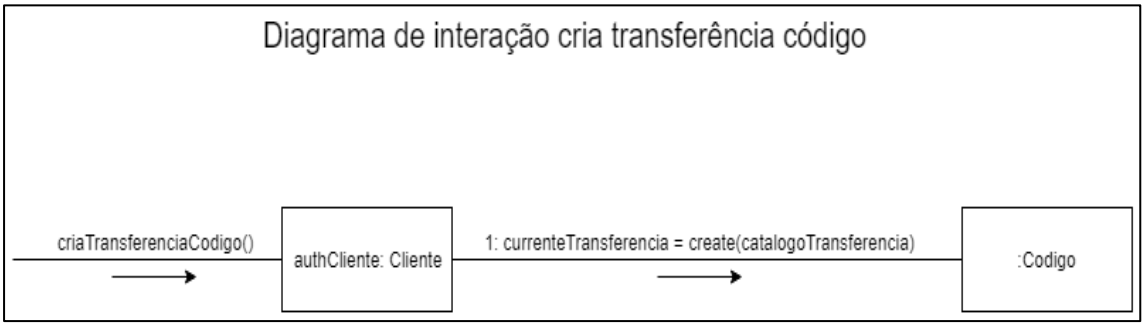
(https://gitlab.com/Henry15/Imagens_da_tese/blob/master/Diagramas_de_interacao/Transferencia/mostra_transferencias_frequentesInterbancarias.png)

E.20 – confirma transferência de uma já existente



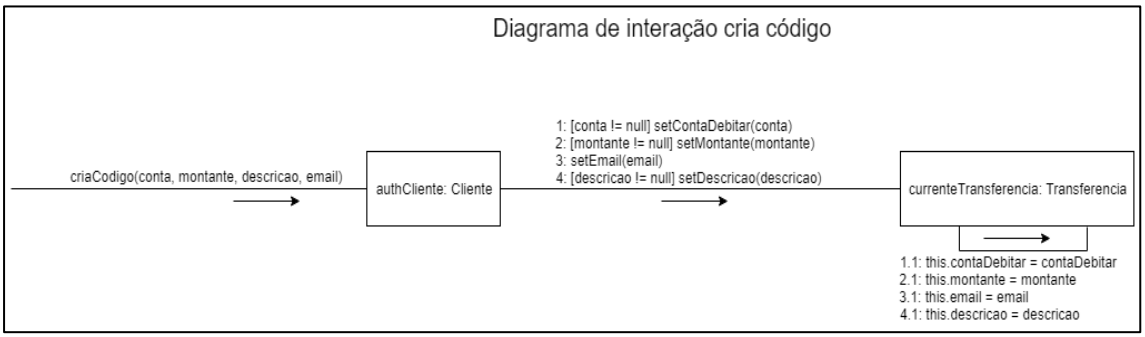
(https://gitlab.com/Henry15/Imagens_da_tese/blob/master/Diagramas_de_interacao/Transferencia/confirma_transferencia_de_uma_ja_existente.png)

E.21 – cria transferência por código



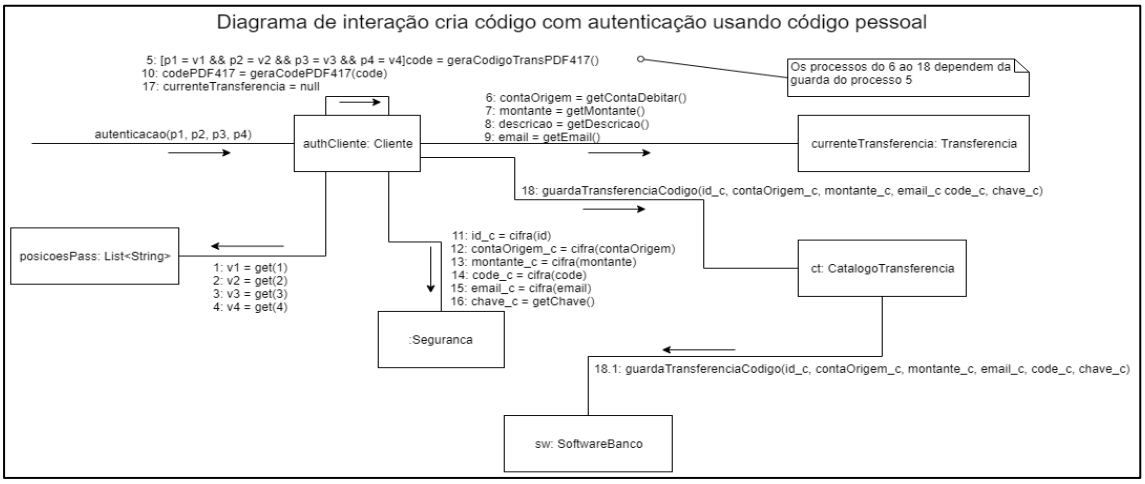
(https://gitlab.com/Henry15/Imagens_da_tese/blob/master/Diagramas_de_interacao/Transferencia/escolhe_transferencia_codigo.png)

E.22 – cria código de transferência



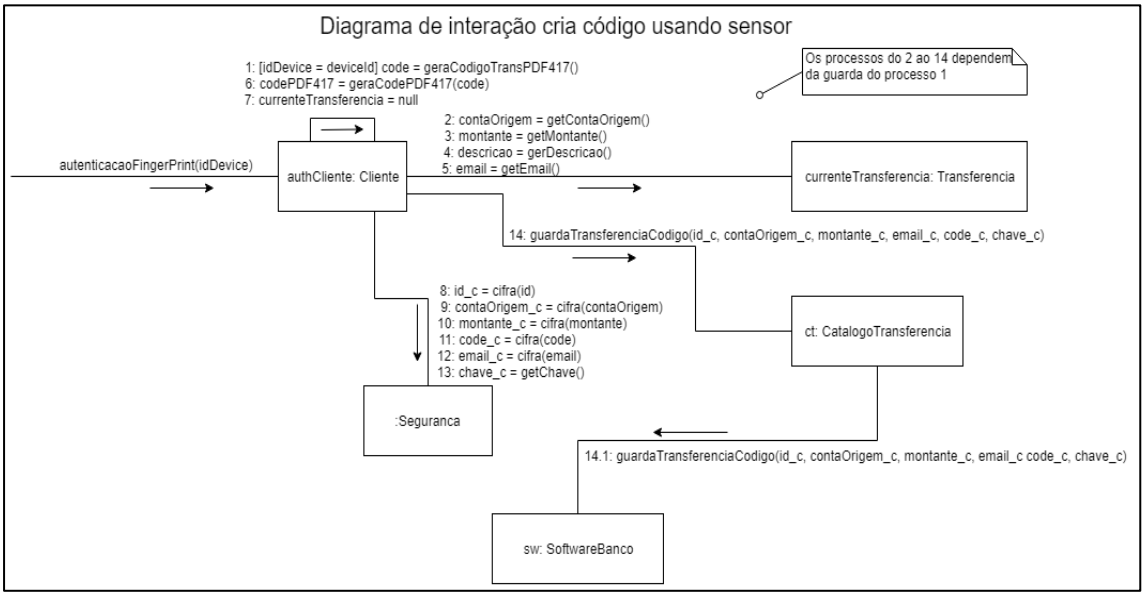
(https://gitlab.com/Henry15/Imagens_da_tese/blob/master/Diagramas_de_interacao/Transferencia/cria_codigo_transferencia.png)

E.23 – cria código de transferência com autenticação usando código pessoal



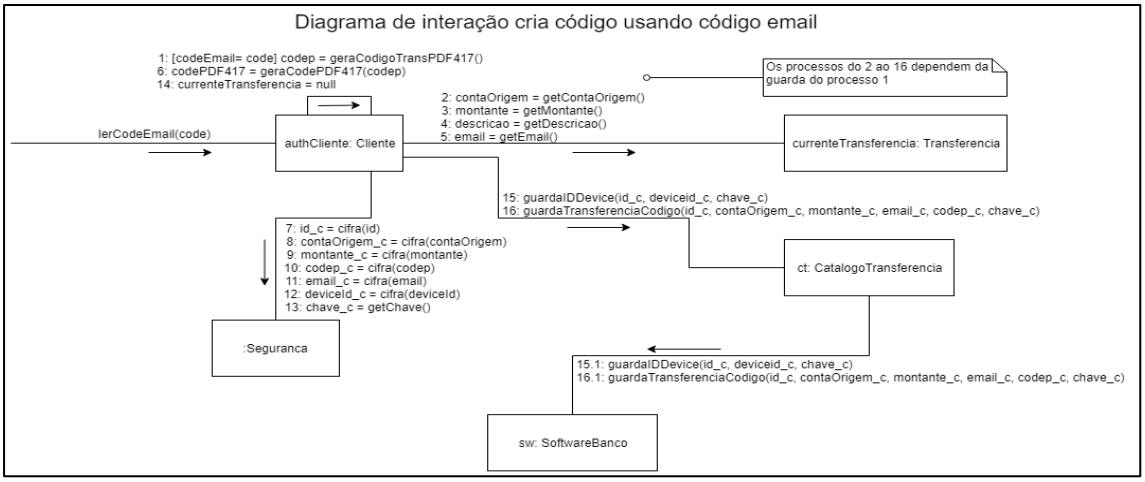
(https://gitlab.com/Henry15/Imagens_da_tese/blob/master/Diagramas_de_interacao/Transferencia/cria_codigo_transferencia_usando_codigo_pessoal.png)

E.24 – cria código de transferência com autenticação usando sensor



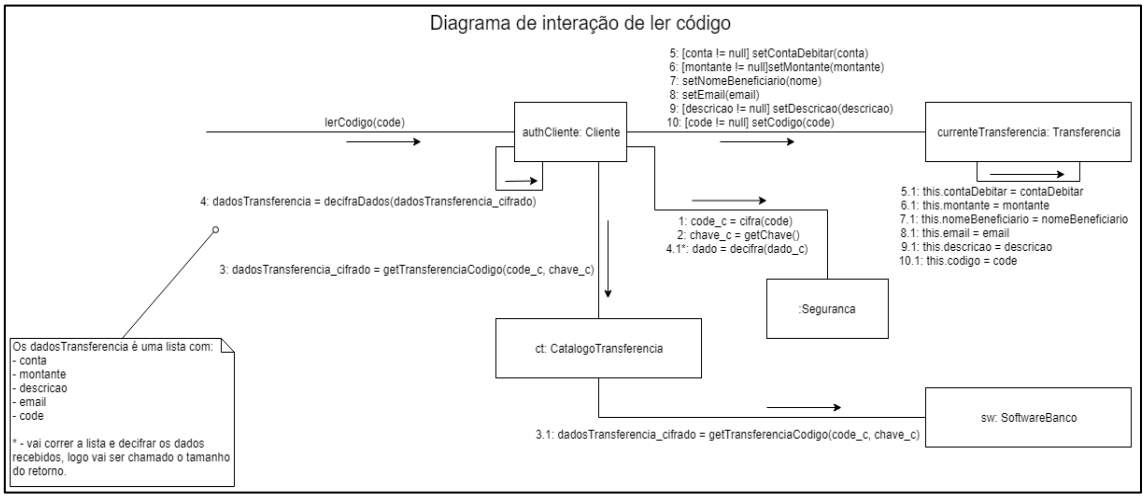
(https://gitlab.com/Henry15/Imagens_da_tese/blob/master/Diagramas_de_interacao/Transferencia/cria_codigo_transferencia_sensor.png)

E.25 – cria código de transferência com autenticação usando código de email



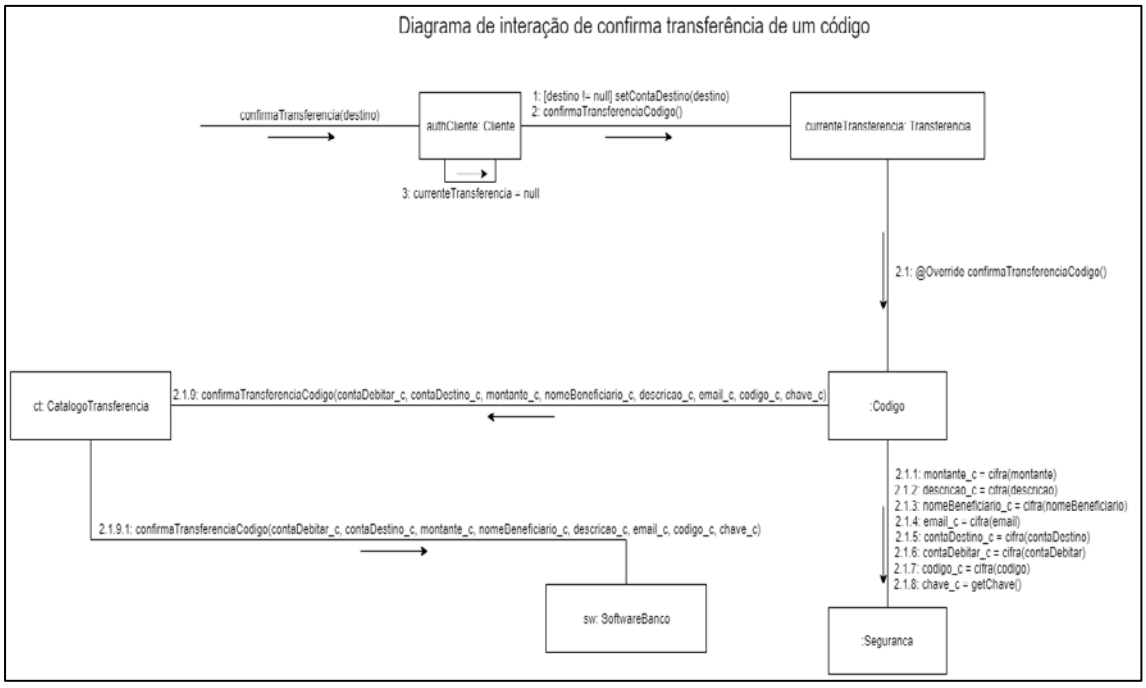
(https://gitlab.com/Henry15/Imagens_da_tese/blob/master/Diagramas_de_interacao/Transferencia/cria_codigo_transferencia_usando_codigo_email.png)

E.26 – leitura de código de transferência



(https://gitlab.com/Henry15/Imagens_da_tese/blob/master/Diagramas_de_interacao/Transferencia/le_codigo_transferencia.png)

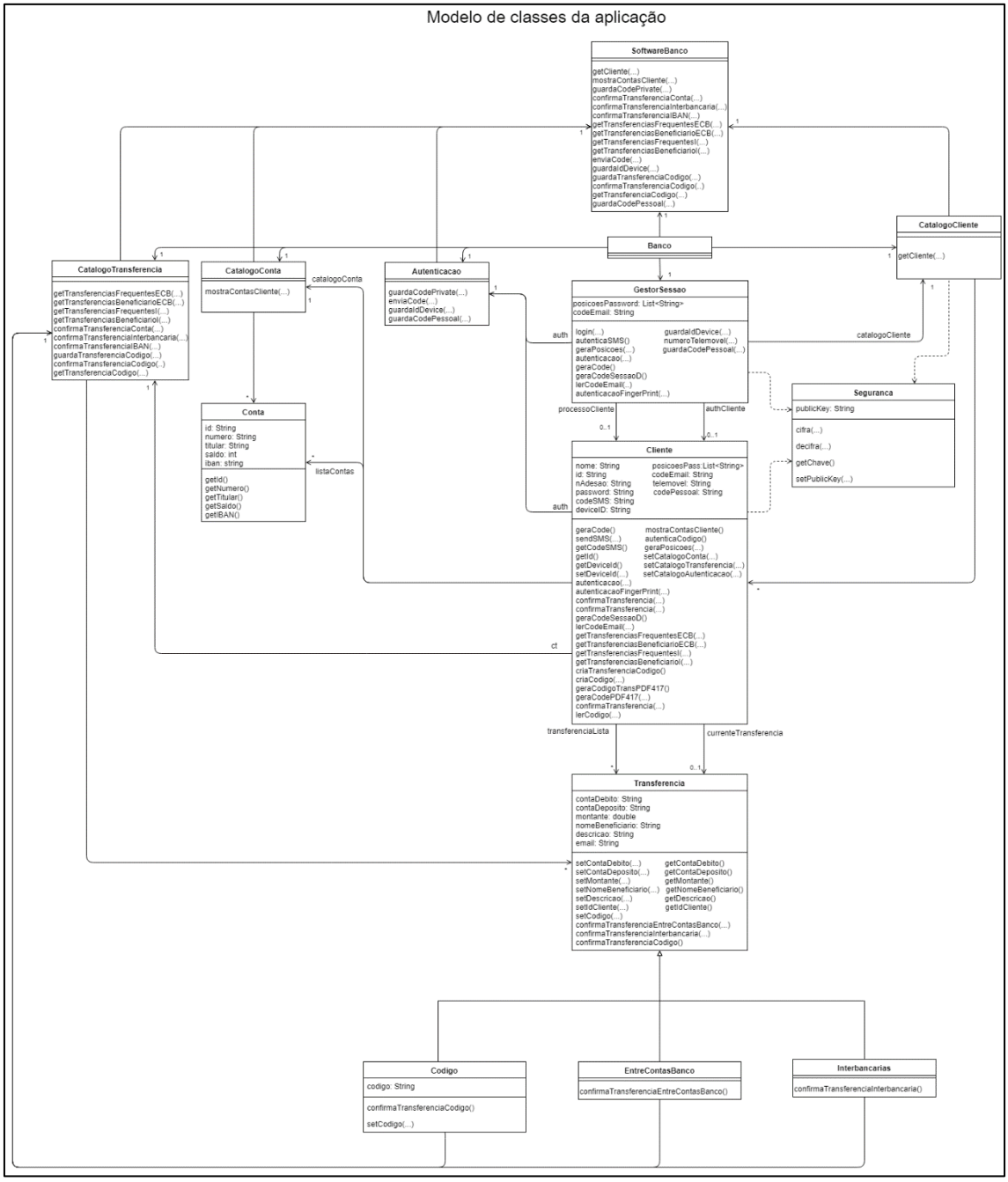
E.27 – confirma a transferência de código



(https://gitlab.com/Henry15/Imagens_da_tese/blob/master/Diagramas_de_interacao/Transferencia/confirma_transferencia_de_um_codigo.png)

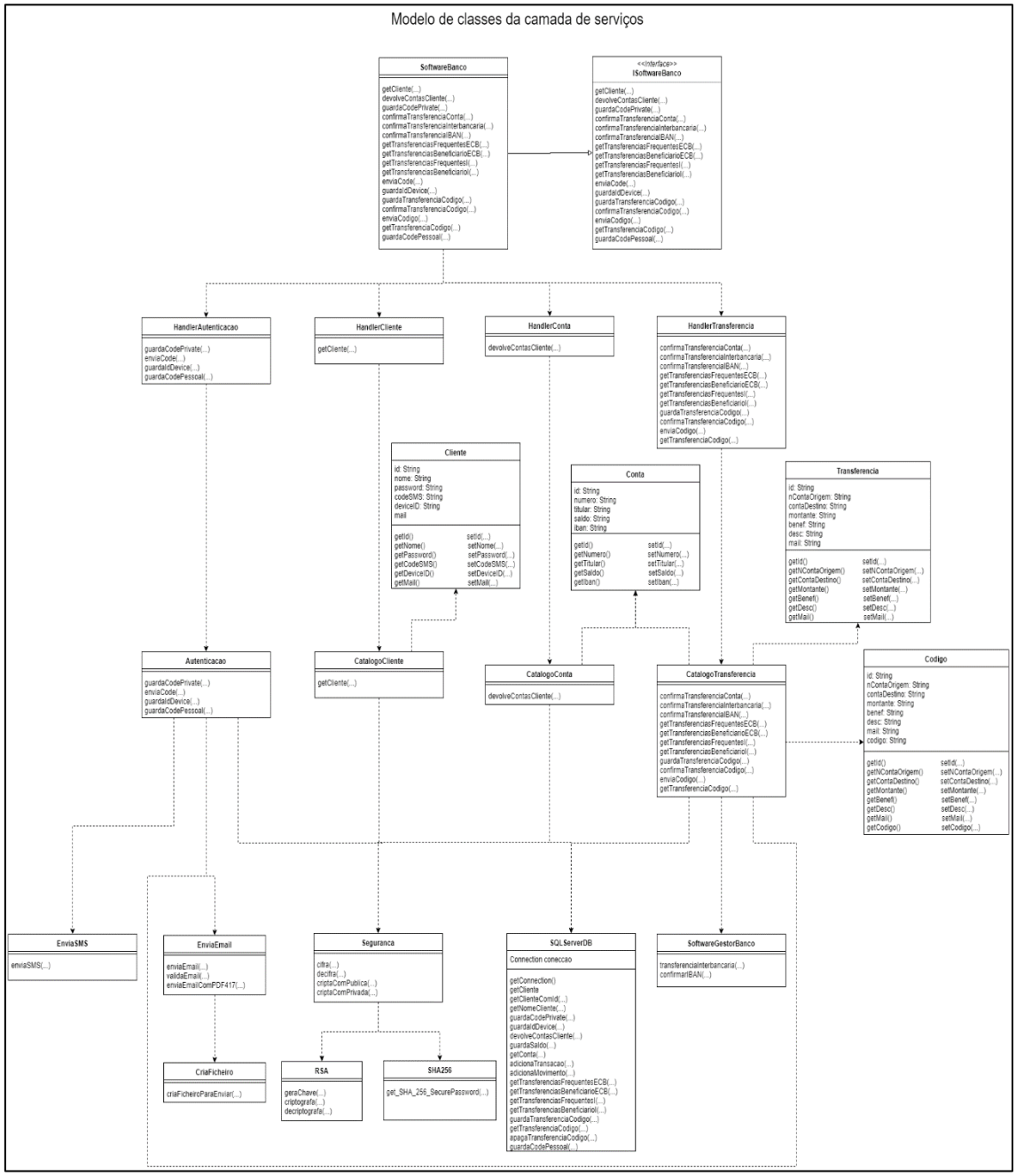
Anexo F – Modelo de classes

F.1 – Modelo de classes da aplicação móvel



(https://gitlab.com/Henry15/Imagens_da_tese/blob/master/Modelo_de_classes/Aplicacao_movel/Modelo_de_classes.png)

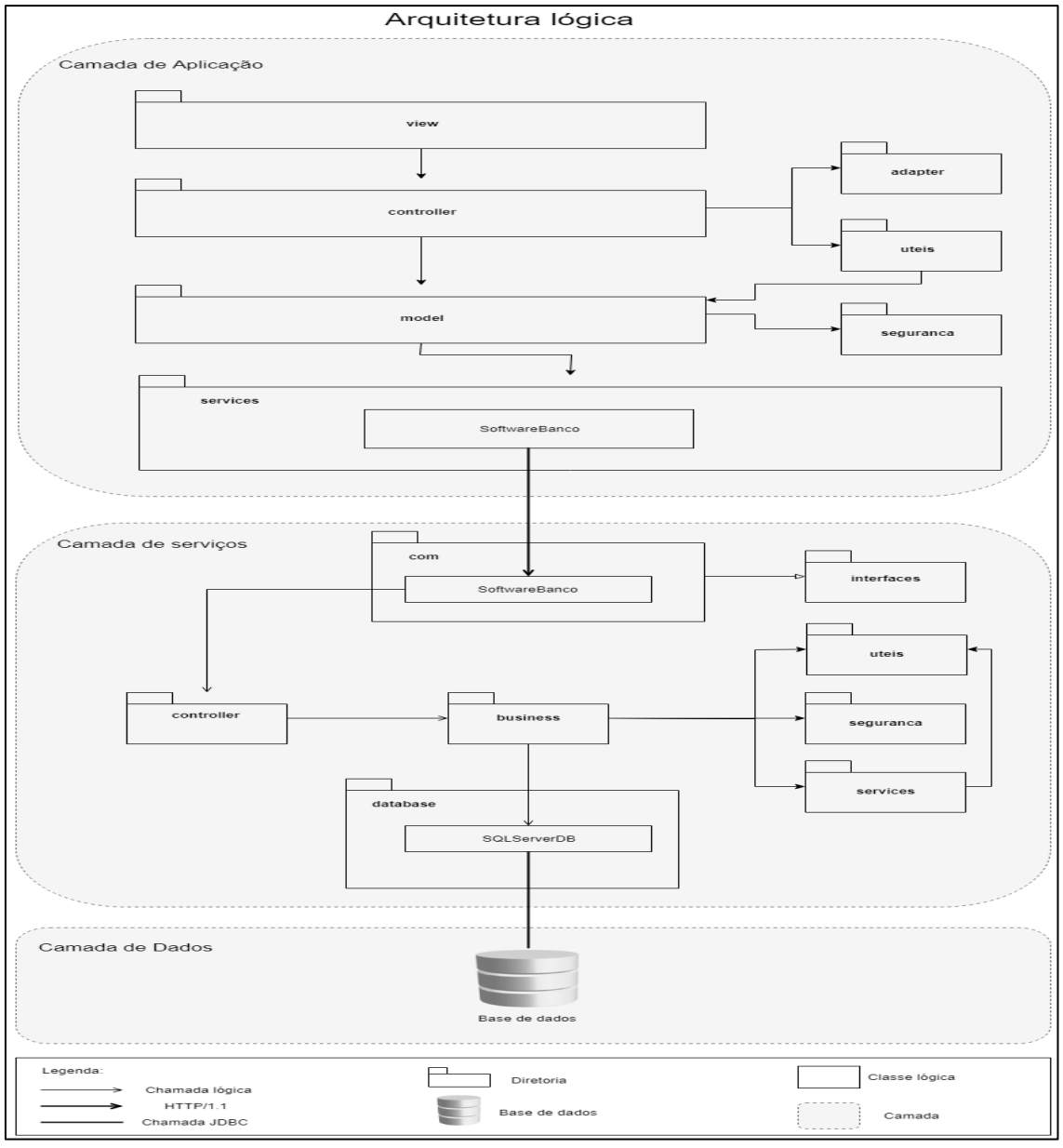
F.2 – Modelo de classes dos serviços



(https://gitlab.com/Henry15/Imagens_da_tese/blob/master/Modelo_de_classes/Servicos/Modelo_de_classes_servico.png)

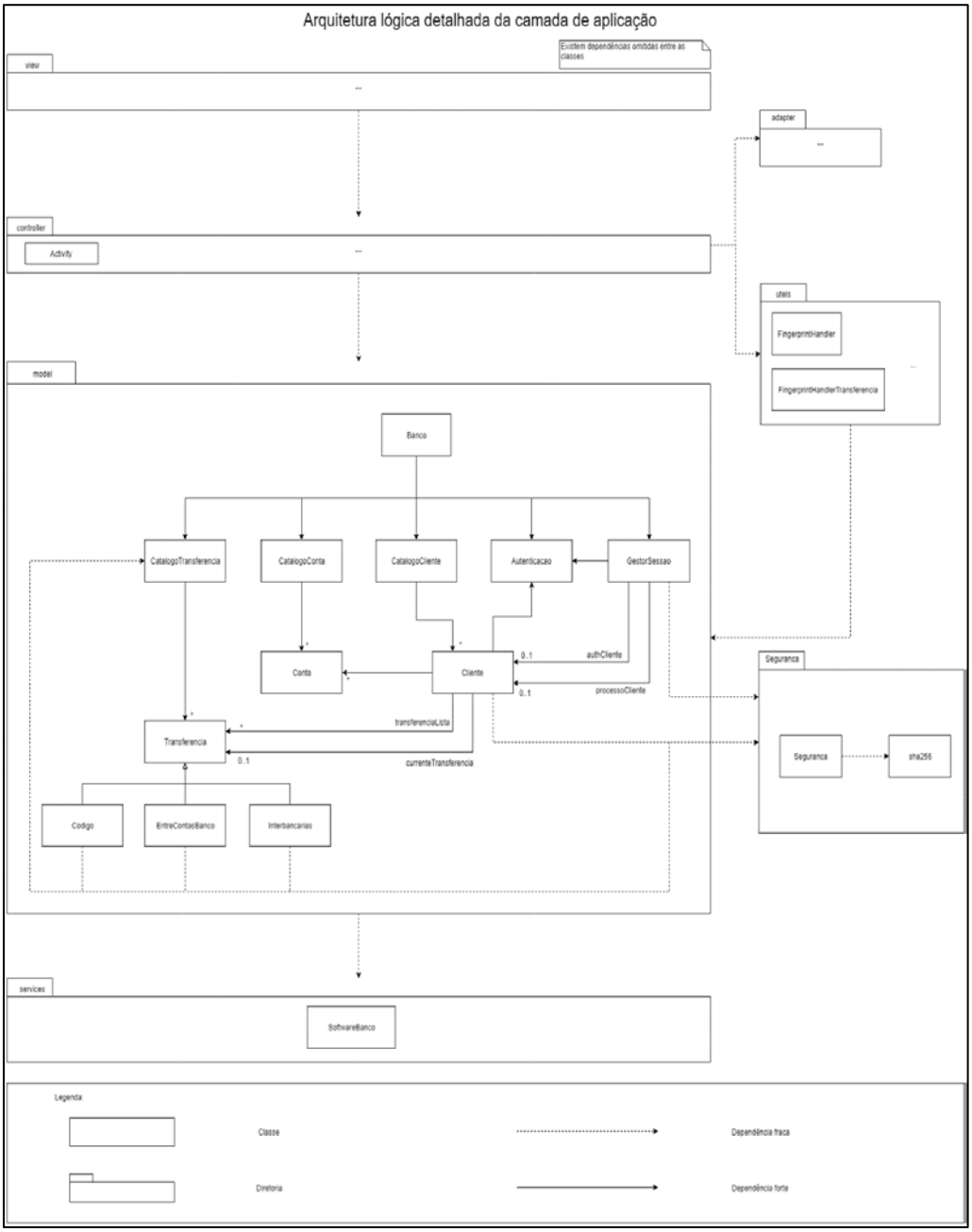
Anexo G – Arquitetura lógica do sistema

G.1 – Arquitetura lógica do sistema



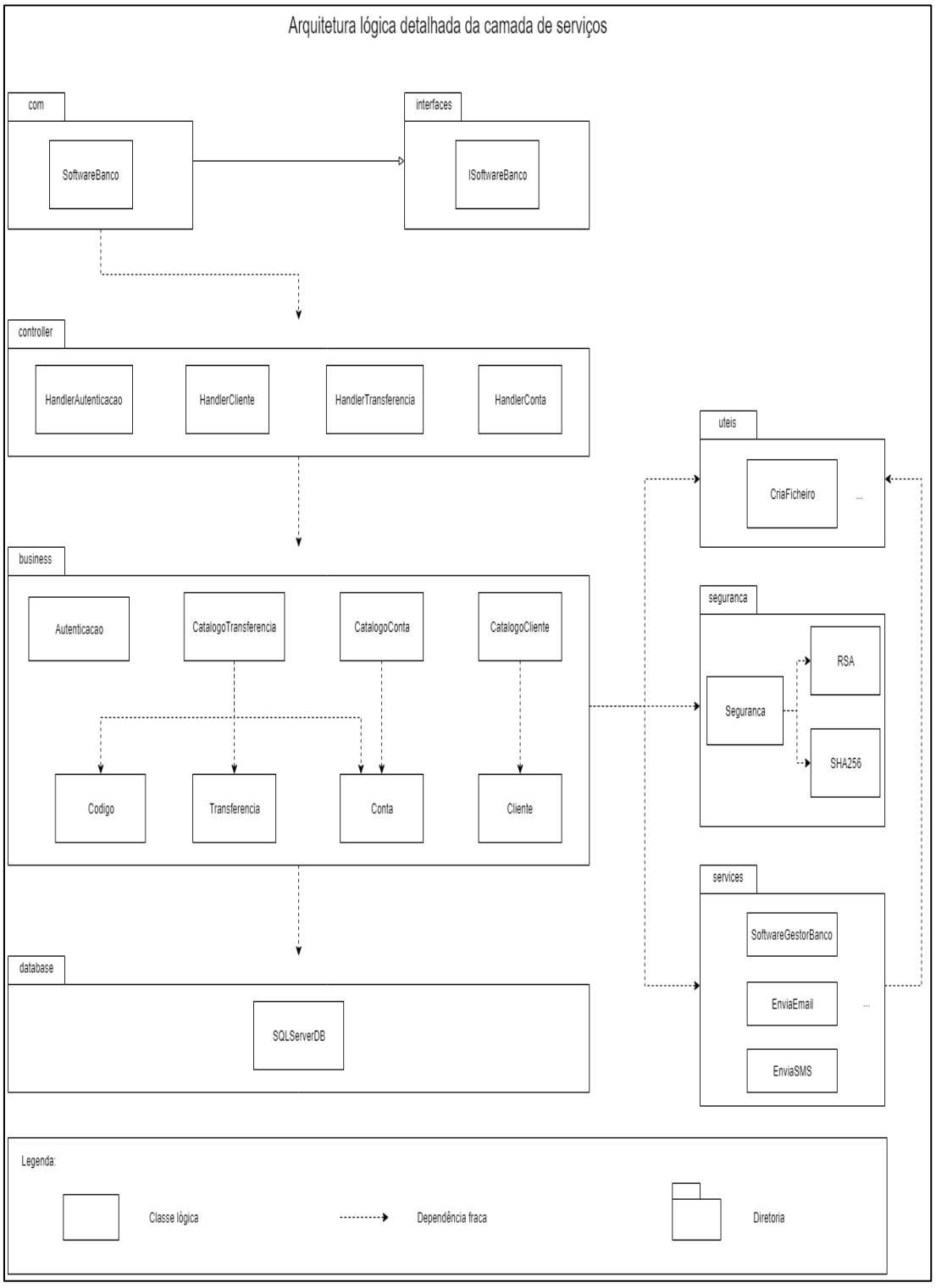
(https://gitlab.com/Henry15/Imagens_da_tese/blob/master/Arquitetura_do_sistema/Logica/Arquitetura_logica.png).

G.2 – Arquitetura lógica detalhada da camada da aplicação móvel



(https://gitlab.com/Henry15/Imagens_da_tese/blob/master/Arquitetura_do_sistema/Logica/Aplicacao_movel/Arquitetura_logica_da_aplicacao.png)

G.3 – Arquitetura lógica detalhada da camada de serviços

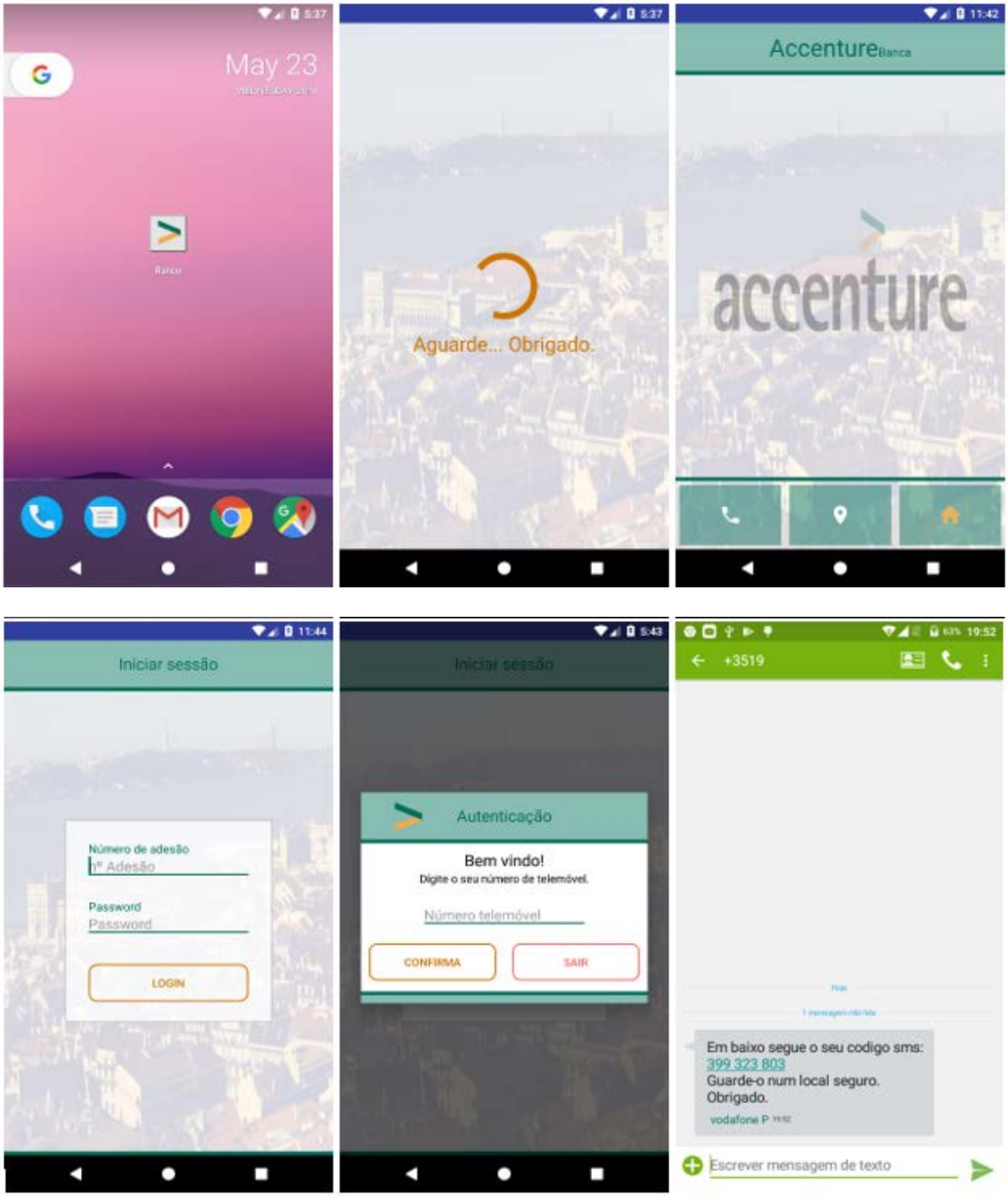


(https://gitlab.com/Henry15/Imagens_da_tese/blob/master/Arquitetura_do_sistema/Logica/Servicos/Arquitetura_logica_dos_servicos.png)

Anexo H – Testes de sistema

H.1 – Teste 1: Inicialização e acesso à conta

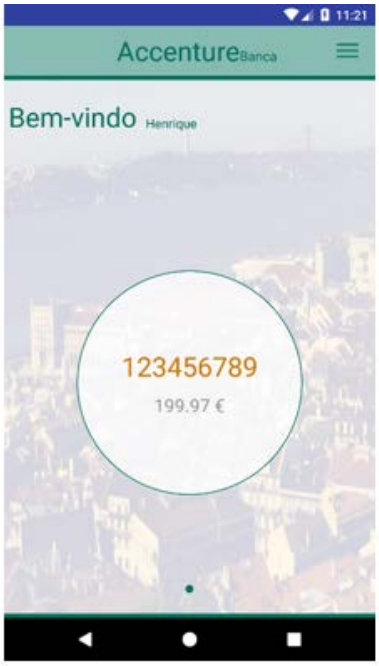
Sequência de interfaces na realização do teste 1:




5:43
Iniciar sessão
Autenticação
Digite um código de 6 números.
Atenção! digite um código forte e guarde-o.
Código SMS: Código SMS
Código Pessoal: Código Pessoal
CONFIRMA
SAIR

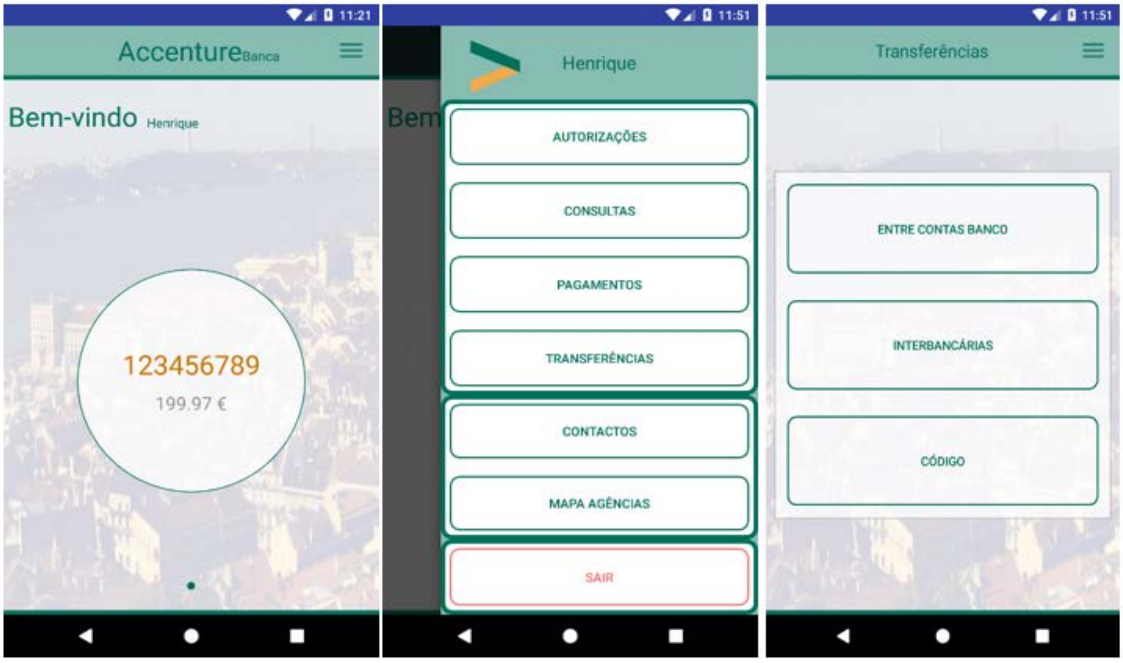

5:55
Iniciar sessão
Autenticação
CÓDIGO PESSOAL
IMPRESSÃO DIGITAL
Introduza as seguintes posições do seu código pessoal:
3º 2º 5º 4º
AUTENTICAÇÃO
SAIR

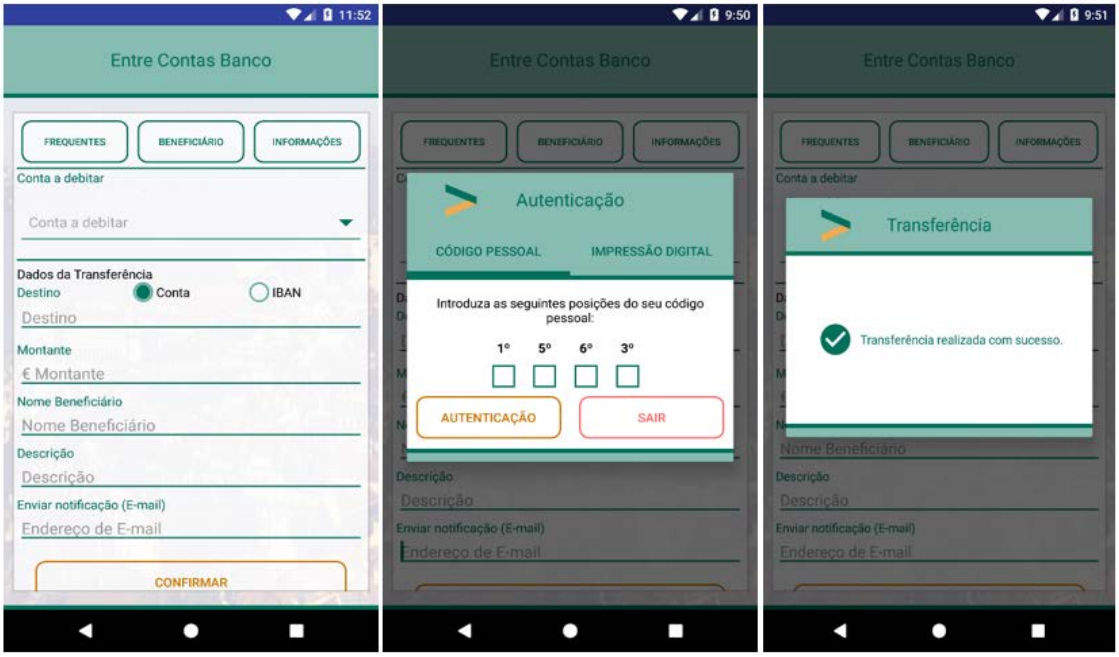

5:46
Iniciar sessão
Autenticação
Foi autenticado com sucesso.


11:21
AccentureBanca
Bem-vindo Henrique
123456789
199.97 €

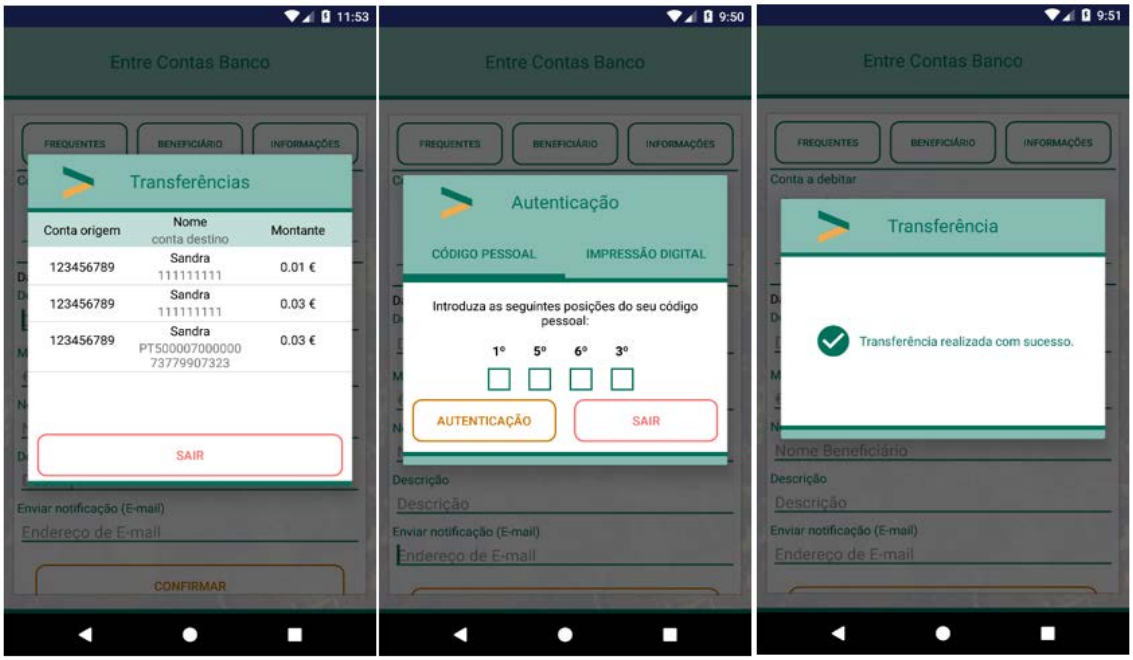
H.2 – Teste 2: Realização de uma transferência entre contas do mesmo banco

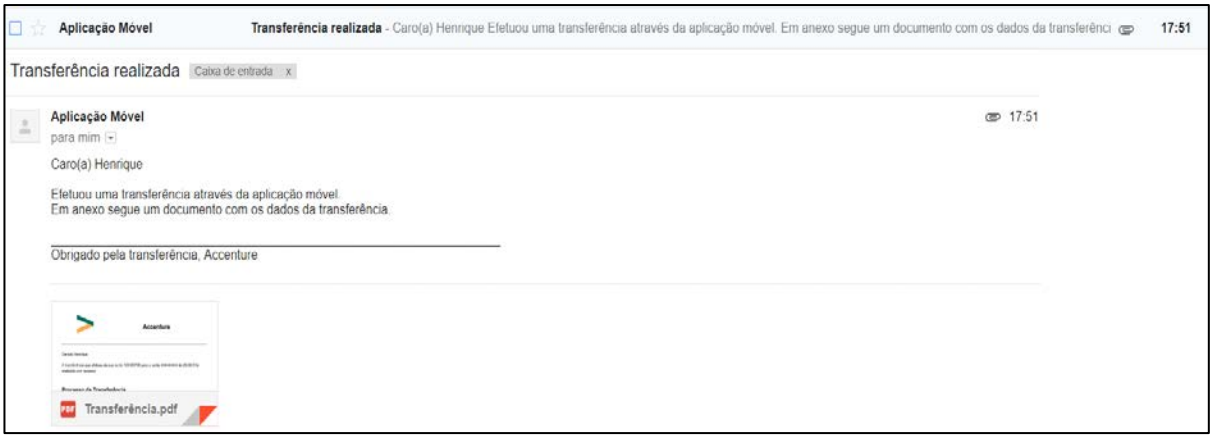
Sequência de interfaces na realização do teste 2:





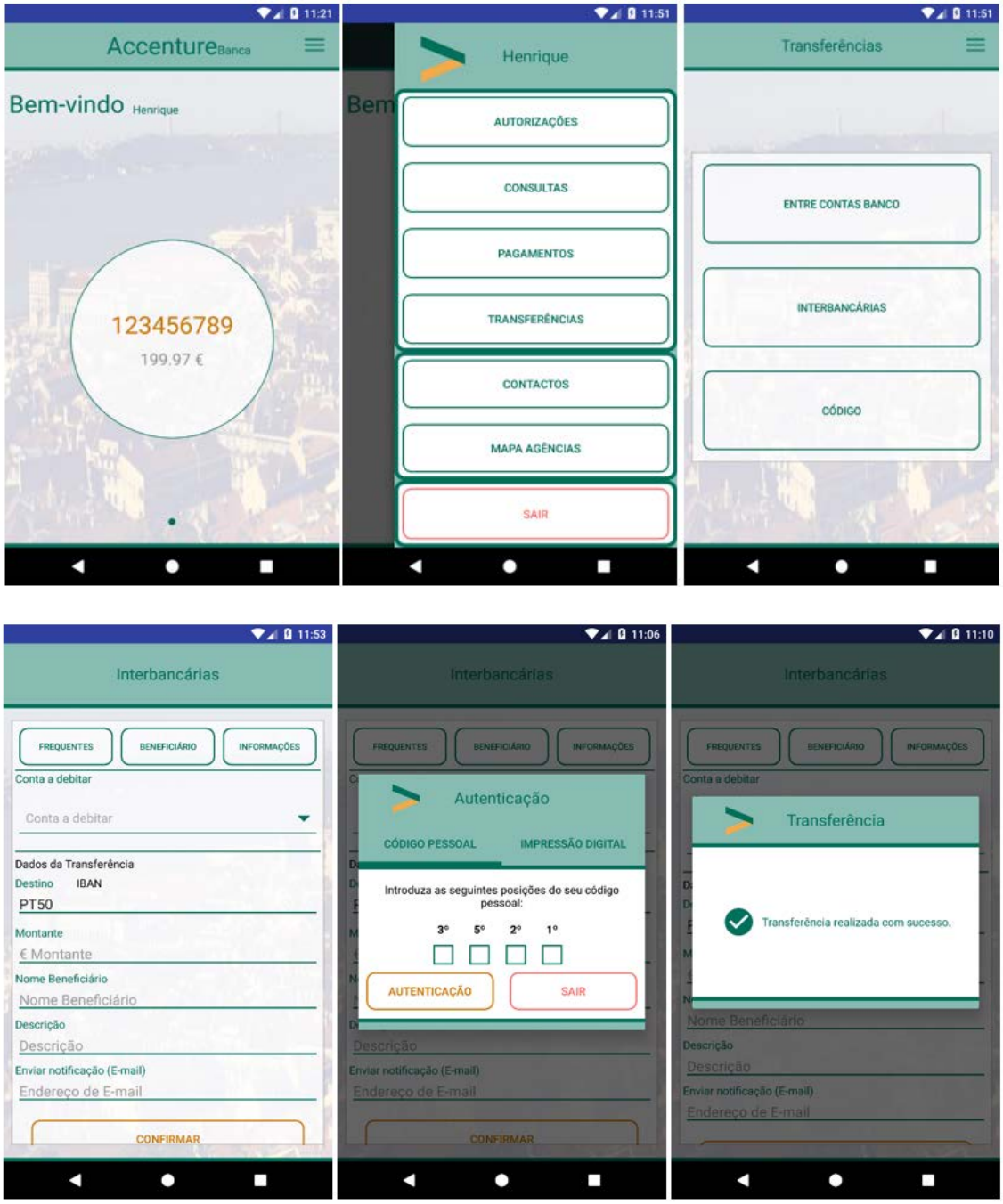
Sequência de interfaces na realização do teste 2 (processo agilizado):



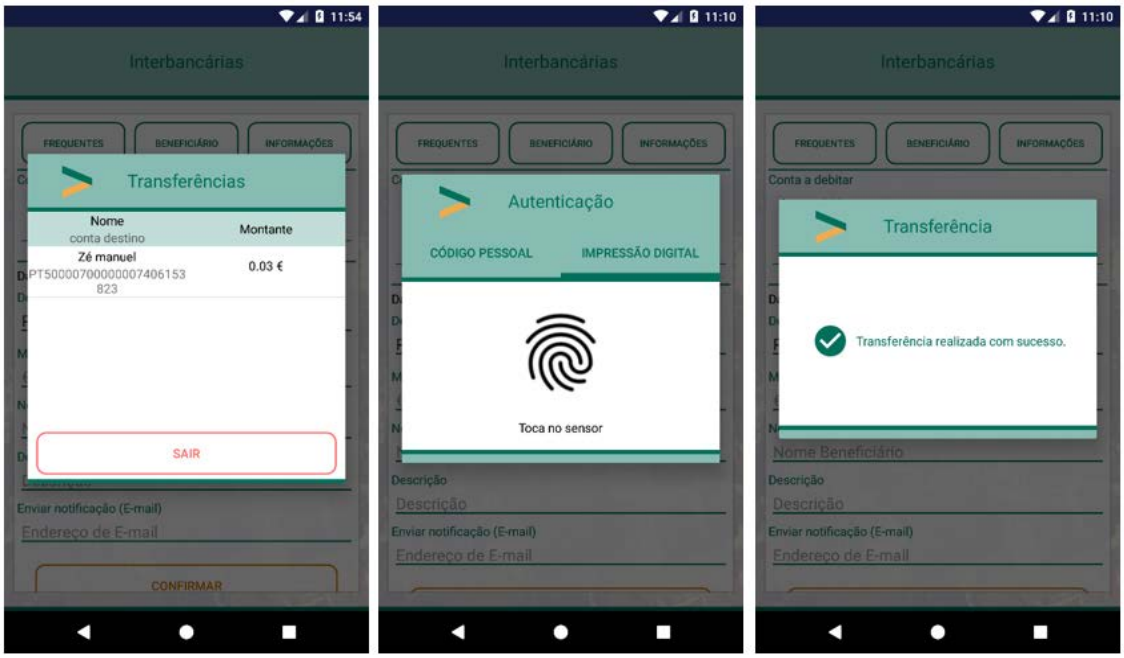


H.3 – Teste 3: Realização de uma transferência interbancária

Sequência de interfaces na realização do teste 3:

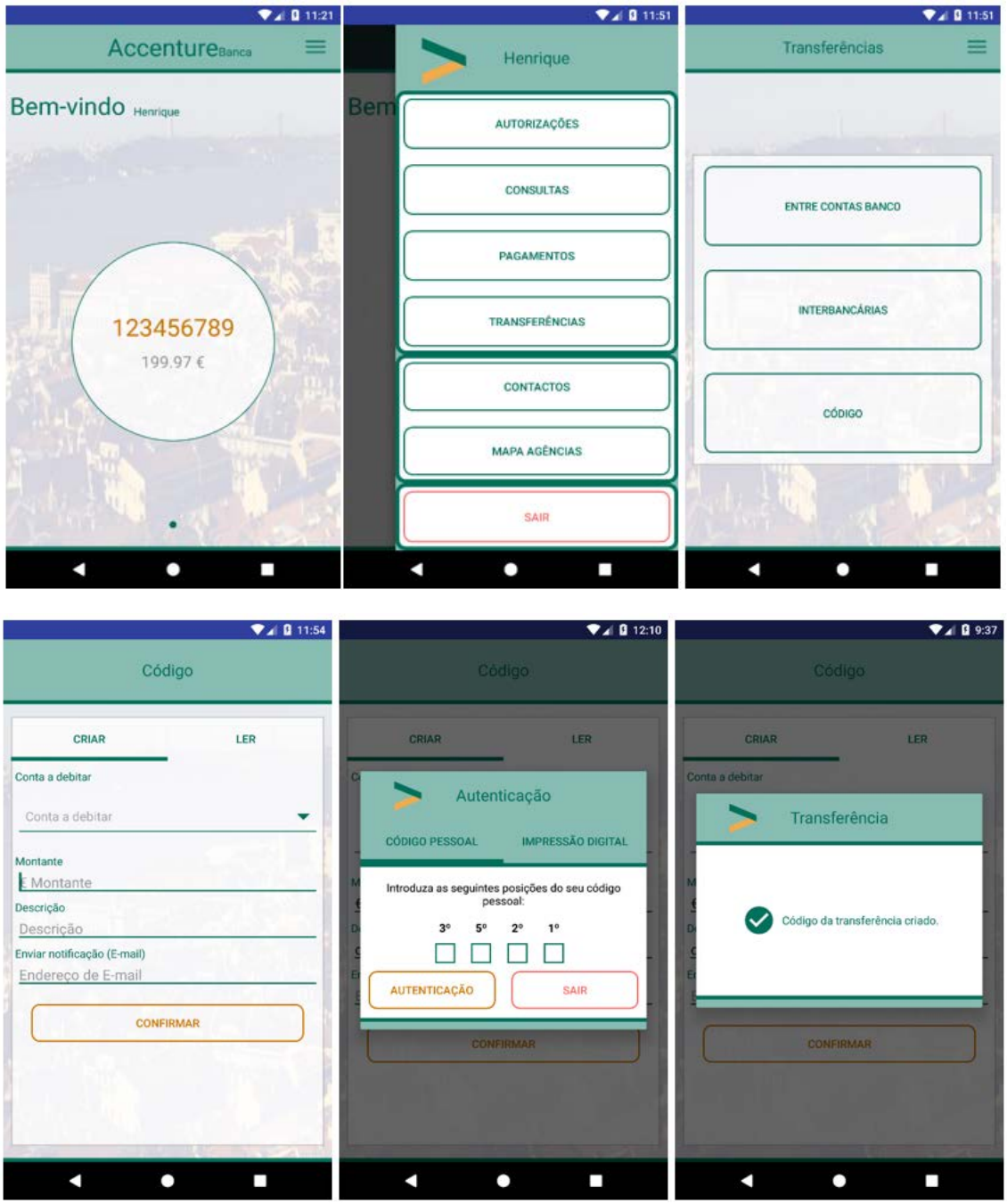


Sequência de interfaces na realização do teste 3 (processo agilizado):

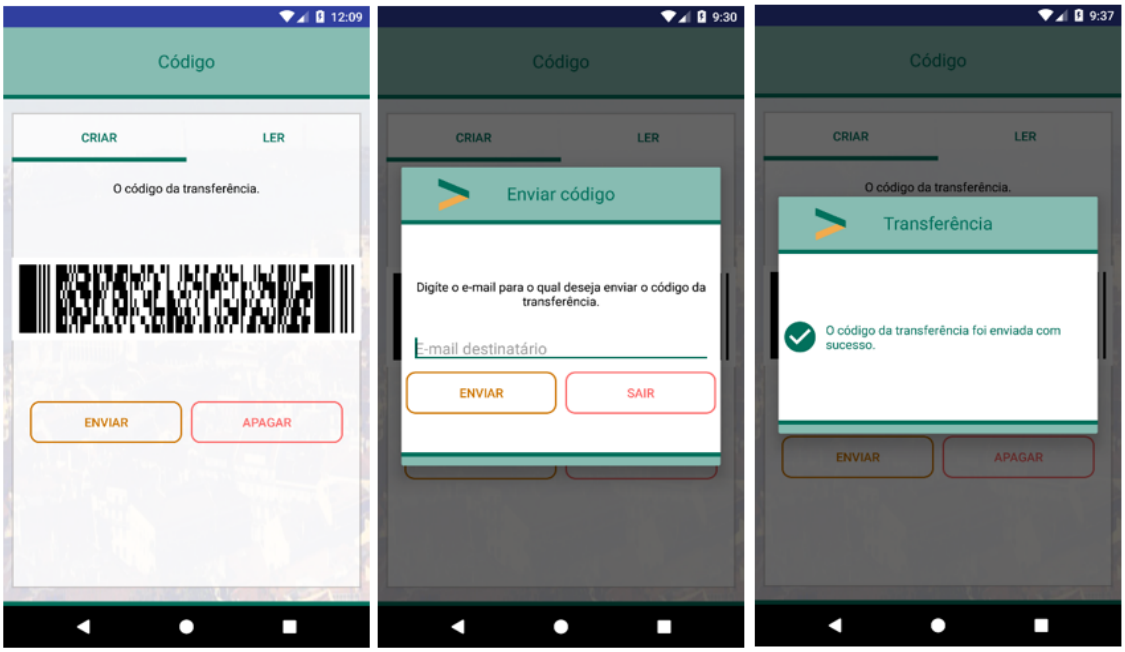


H.4 – Teste 4: Criação de um código de transferência

Sequência de interfaces na realização do teste 4:



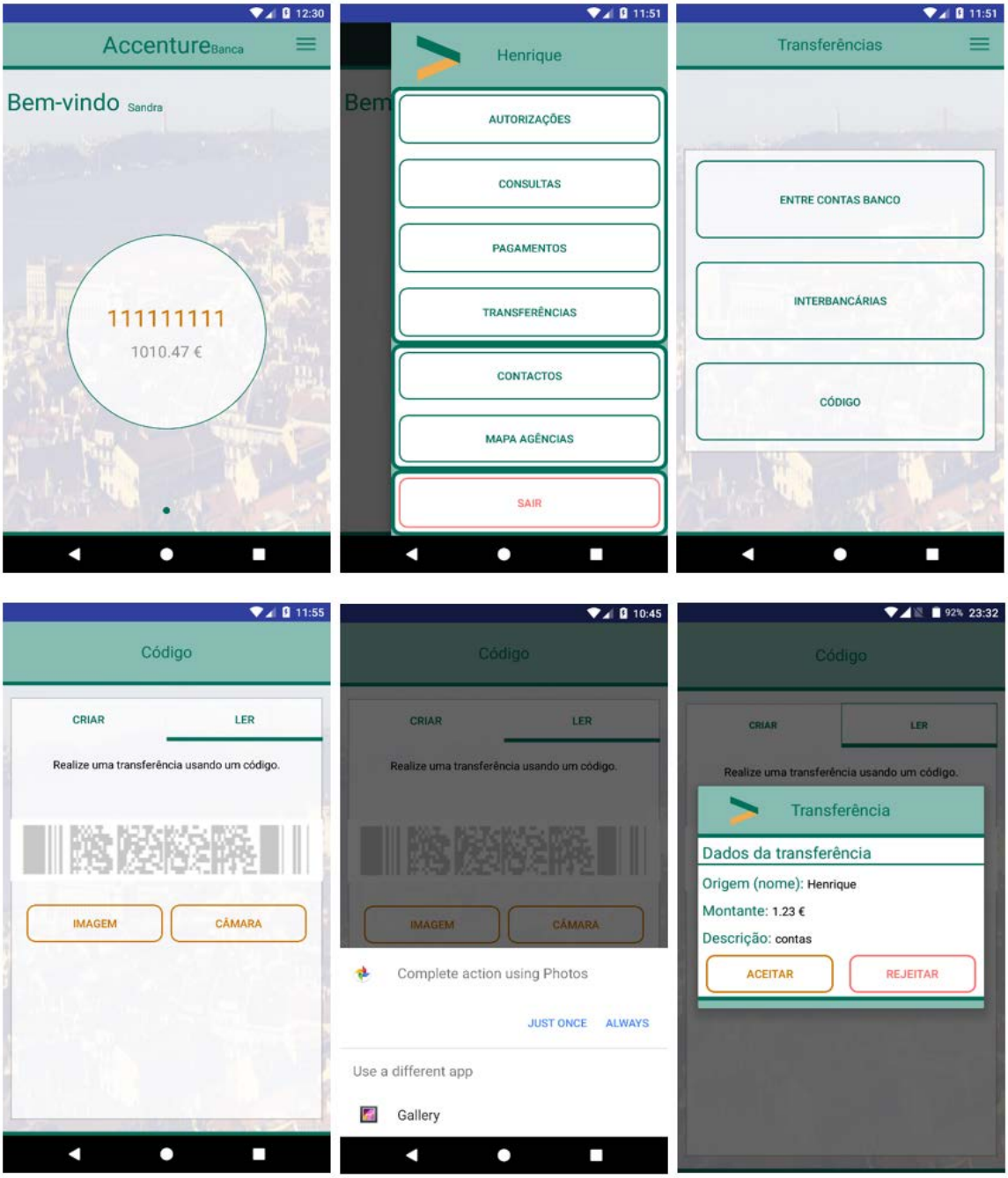
Sequência de interfaces na realização do teste 4 (envio do código de transferência por email):

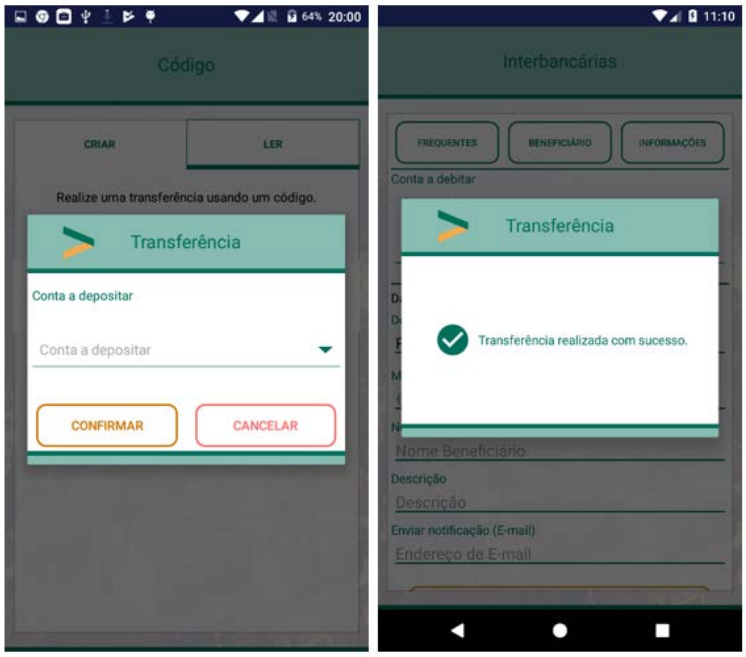


Sequência de interfaces na realização do teste 4 (apagar código de transferência):

H.5 – Teste 5: Leitura e realização de uma transferência por código

Sequência de interfaces na realização do teste 5:




64% 20:00
Código
CRIAR
LER
Realize uma transferência usando um código.
Transferência
Conta a depositar
Conta a depositar
CONFIRMAR
CANCELAR
11:10
Interbancárias
FREQUENTES
BENEFICIÁRIO
INFORMAÇÕES
Conta a debitar
Transferência
Transferência realizada com sucesso.
Nome Beneficiário
Descrição
Descrição
Enviar notificação (E-mail)
Endereço de E-mail